Universidad de las Ciencias Informáticas

Facultad 7

Componente para la creación y manipulación de implantes digitales ortopédicos

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Autor: Sergio Manuel Pérez Mayo

Tutores: Ing. Jesmar Ernesto Fajardo Martín

Ing. Filiberto López Palenzuela

La Habana, Junio del 2010

"Año 53 de la Revolución."

DATOS DE CONTACTO

Tutores:

Tutor principal:

Ing. Jesmar E. Fajardo Martín (jefajardo@uci.cu):

Graduado de Ingeniero en Ciencias Informáticas, egresado de la UCI en el 2008. Ha impartido las asignaturas de Programación II, Compiladores, Inteligencia Artificial y Práctica Profesional 2. Es profesor Instructor de la Facultad 7 y se desempeña actualmente como Programador en el Departamento de Producción de Software Médico Imagenológico, CESIM.

Co tutor

Ing. Filiberto López Palenzuela (flopez@uci.cu):

Graduado de Ingeniero en Ciencias Informáticas, egresado de la UCI en el 2009. Ha impartido las asignaturas de Programación II y Práctica Profesional 4. Es profesor instructor de la Facultad 7 y se desempeña actualmente como Arquitecto en el Departamento de Producción de Software Médico Imagenológico, CESIM.

DEDICATORIA

A la Revolución y a Fidel por hacer posible que exista ésta Universidad de excelencia.

A mis padres, por su dedicación, apoyo, preocupación y cariño.

Al Dr. Carlos Gassiot por su apoyo incondicional y oportunos consejos.

A mis amistades por estar siempre ahí cuando las necesito y por formar parte de mi vida…

A Jesmar y Filiberto por ser mis tutores, a todas las personas que de una forma u otra me ayudaron con sus críticas y consejos.

A todas las personas que han contribuido de alguna manera, a mi formación profesional a través de sus enseñanzas y lecciones.

AGRADECIMIENTOS

A mis padres.

Por haberme enseñado todo lo que soy como persona, por haberme inculcado los principios, los valores, la perseverancia, la dedicación y el empeño, por haberme dado tanto amor, cariño y su apoyo incondicional sin pedir nunca nada a cambio.

A mis familiares.

A mis hermanos que a pesar de estar distantes, siempre me ayudaron y aconsejaron. A mis abuelos por quererme y mimarme tanto, por ayudarme y apoyarme siempre que lo necesité. A mis sobrinitos Ronni Alejandro y Melissa Daimé por ser especiales para mí.

A mis amigos.

A todos mis amigos de la universidad con los que he compartido clases, fiestas, juegos y actividades en general todos estos años. Al grupo especial de amigos que nos consideramos primos, por siempre estar presentes en las buenas y en las malas.

Sergio Manuel Pérez Mayo.

RESUMEN

La presente investigación surge en el marco de trabajo del Proyecto: “Planificador Quirúrgico Ortopédico”, como parte del proceso de informatización de distintos sectores, entre ellos el de la salud, que se está llevando a cabo en la UCI. En este se realizará la implementación de un componente para la creación y manipulación de implantes digitales ortopédicos, indispensable para el proceso de planificación preoperatoria ortopédica en los casos que se requieran implantes.

El componente constará de dos paquetes, el editor de implantes y la librería de implantes. La aplicación de edición de implantes permitirá diseñar gráficamente las imágenes correspondientes al implante ortopédico así como la información referente al mismo. El uso de diversos patrones permitió que las herramientas de diseño se acoplen al sistema dinámicamente, obteniéndose una aplicación extensible. La librería de implantes, diseñada a modo de catálogo, permitirá manipular y visualizar los implantes seleccionados a través de determinados criterios de búsqueda, así como trasladar y rotar el implante digital ortopédico sobre otra imagen; similar a la planificación quirúrgica manual llevada a cabo mediante la realización de calcos preoperatorios. Esperando obtener como resultado un componente informático que le brinde al Proyecto: “Planificador Quirúrgico Ortopédico”, las funcionalidades necesarias para manipular los implantes digitales esenciales en el proceso de planificación preoperatoria.

Palabras claves:

Componente, planificación quirúrgica ortopédica, implante ortopédico, transparencia de implante, implante digital ortopédico.

TABLA DE CONTENIDOS

Introducción

El sistema de salud cubano es universal, gratuito y accesible a todos los ciudadanos, lo cual se manifiesta en su red de unidades asistenciales en el territorio nacional y su sistema de primero, segundo y tercer nivel de atención médica. En ellos se desarrollan diferentes programas priorizados, en cuyo centro de interés sitúan por igual el cuidado del niño, la madre, la mujer y el adulto mayor. Así como la prevención y control de las enfermedades transmisibles y no transmisibles que puedan afectar a los cubanos.

El sistema de salud cubano se basa en el modelo de medicina familiar y en la garantía de accesibilidad a los servicios de salud de manera universal y gratuita, partiendo de los principios y características sostenidos por la Salud Pública durante más de 50 años. Ofrece los adelantos de la ciencia y la técnica de que dispone el país, sin distinción de política, raza o religión. Igualmente se asienta sobre bases jurídicas y se rige según su estructura jerárquico administrativa territorial.

Dada las políticas aplicadas por el MINSAP en el campo de la medicina y el desarrollo científico y técnico, según, (1) implicará que Cuba tendrá la población más envejecida de Latinoamérica en los albores del año 2025, y para el 2050 se espera que los cubanos tengan uno de los promedios de edad más elevados del planeta, según pronósticos de expertos en demografía. El envejecimiento de la población de la isla está asociado al aumento de la esperanza de vida del pueblo cubano que, con 77 años, es muy similar a la de los países desarrollados. Lo que trae como consecuencia un mayor número de pacientes con afectaciones en el sistema osteomioarticular comunes en personas de avanzada edad.

En el amplio sistema de salud que posee Cuba, se encuentra enmarcado el complejo hospitalario más extenso e integral del mundo, dedicado a la cirugía ortopédica, traumatología, reconstrucción y rehabilitación del sistema osteomioarticular, el Hospital Ortopédico "Frank País". Dirigido por el Prof. Dr. Sc. Rodrigo Álvarez Cambra, eminente científico cubano, quien brinda toda su experiencia en el campo de la ortopedia, junto a un equipo multidisciplinario de alto nivel científico y ético, que logra un rápido y eficaz restablecimiento en múltiples afecciones. (2)

La planificación preoperatoria ortopédica, es un procedimiento indispensable que debe realizarse previo a la intervención quirúrgica y cuyos objetivos son: determinar el resultado final de la cirugía y establecer la táctica quirúrgica a seguir. Según varios procedimientos, la planificación preoperatoria para una cirugía plantea la realización de calcos preoperatorios, como se muestra en la figura 1.

Figura 1. Calco preoperatorio.

Los calcos permiten entender la complejidad de la fractura, la forma de reducción de la misma, la aplicación del principio biomecánico y la elección del implante necesario. En la cirugía para la colocación de una prótesis de cadera, por ejemplo, es importante conocer ciertos factores previos: tipo y tamaño de la prótesis, posición y orientación correcta de los implantes, tamaño del acetábulo y del hueso; y así lograr reducir el tiempo de la cirugía y obtener resultados satisfactorios.

Los implantes ortopédicos se utilizan para reemplazar las articulaciones dañadas o con problemas y se realiza solamente por cirujanos altamente especializados y capacitados. Cada procedimiento de implante consiste en la extracción de la articulación dañada y un reemplazo de prótesis artificial. Los implantes ortopédicos son construidos principalmente de aleaciones de titanio para la fuerza y forrada con plástico para que actúe como cartílago artificial. Algunos son cementados en su lugar y los demás son presionados a la medida para permitir que el hueso crezca en el implante para la fuerza. Los implantes ortopédicos están disponibles para la cadera, rodilla, hombro y codo.

La necesidad primaria de los implantes ortopédicos es el resultado de la osteoartritis. Cada implante está diseñado para resistir el movimiento y el estrés asociado a cada conjunto individual y para proporcionar una mayor movilidad y la disminución del dolor. Los implantes son sugeridos como una opción, sólo si todos los tratamientos no quirúrgicos han fracasado, incluyendo la pérdida de peso. (3)

En la actualidad la planificación quirúrgica ortopédica en el Hospital Ortopédico Frank País, se realiza de forma manual utilizando transparencias de implantes ortopédicos en caso de que se requiera uno de

estos. Estas transparencias pueden deteriorarse o perderse, además de que su uso es incómodo e introduce errores humanos. Hoy en día los estudios imagenológicos son de naturaleza digital y basados en el estándar DICOM, de allí que estas transparencias de implantes se convierten en una herramienta obsoleta para la planificación quirúrgica ortopédica.

Una de las soluciones que se presentan actualmente para este problema, son los implantes digitales ortopédicos, los cuales no poseen un formato estandarizado y no existe bibliografía que referencie acerca de la correcta manipulación y visualización. Esto constituye un inconveniente para compartir los mismos entre sistemas de planificación quirúrgica ortopédica, además de que por lo general dichos sistemas no exponen información sobre el proceso de gestión de implantes digitales ortopédicos.

Por lo antes planteado se identifica como **problema a resolver:** ¿Cómo estandarizar la manipulación y visualización de los implantes digitales ortopédicos para la planificación quirúrgica ortopédica?

Este problema se enmarca en el **objeto de estudio**: Los procesos para la planificación quirúrgica ortopédica.

El objeto delimita el **campo de acción**: El proceso de gestión de implantes digitales ortopédicos.

Para la solución del problema se plantea como **objetivo general**: Desarrollar un componente informático que centre, dentro del Proyecto Planificador Quirúrgico Ortopédico, la creación y manipulación de implantes digitales ortopédicos.

Para dar cumplimiento al objetivo planteado se proponen las siguientes **tareas de la investigación**:

1. Evaluar las tendencias actuales en construcción de librerías de software.
2. Identificar las buenas prácticas o los procedimientos estándares en la manipulación de las librerías de implantes digitales ortopédicos.
3. Seleccionar las herramientas y tecnologías existentes más adecuadas para la manipulación de implantes digitales ortopédicos.
4. Analizar los procesos de negocio asociados a la gestión de los implantes digitales ortopédicos, logrando un modelo único como guía para la implementación del sistema.
5. Generar los artefactos correspondientes a los Flujos de Trabajo propuestos por la Metodología seleccionada, sirviendo de base a los desarrolladores.

6. Implementar el sistema informático aplicando las pautas de diseño y siguiendo lo establecido en la Especificación de Requisitos de Software.

Como **resultado esperado** se obtendrá un componente de software que permita diseñar implantes digitales ortopédicos y almacenar los mismos en un formato flexible. Lo cual permitirá su ulterior uso y visualización por el proyecto planificador quirúrgico brindándole así una función esencial para realizar el proceso de planificación quirúrgica.

Estructura de los capítulos

Capítulo 1: Fundamentación teórica. Contiene el estudio del arte sobre las librerías y editores de implantes digitales ortopédicos, necesarios para una correcta selección de las herramientas y la metodología de desarrollo a utilizar en el desarrollo del componente informático que dé solución a la situación problémica planteada.

Capítulo 2: Características del sistema. Contiene la descripción del sistema basándose en el modelo de dominio, los requisitos funcionales y no funcionales identificados para el diseño e implementación del componente informático.

Capítulo 3: Análisis y diseño. Este capítulo describe cómo implementar el componente, a través del diseño, enfocado a cómo el sistema cumple sus objetivos teniendo en cuenta los requisitos funcionales y no funcionales. Se realizan los diagramas de clases y los diagramas de interacción según los casos de uso definidos y se explica además la arquitectura utilizada, así como el empleo de los patrones de diseño.

Capítulo 4: Implementación del sistema. En este capítulo se describe cómo fue implementado el sistema. Se presenta el diagrama de componentes, que muestra la distribución física de los componentes para la implementación. Se describen los fragmentos relevantes del código.

CAPÍTULO 1. **Fundamentación teórica**

En este capítulo se aborda el tema de las librerías de software y específicamente librerías de implantes digitales ortopédicos, sobre software para la gestión y edición de imágenes digitales; haciendo un estudio del arte de estos mismos en el área de la Ortopedia. Se hace mención a tecnologías, marcos de trabajo, patrones de diseño, metodologías de desarrollo, así como herramientas. De todos ellos se plantea conocer las características que justifiquen su selección para el desarrollo de la solución.

1.1. Librerías de software

En ciencias de la computación, una biblioteca (o librería) es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables. (4)

1.2. Librerías de implantes digitales ortopédicos

En la actualidad existen software de planificación quirúrgica que utilizan una librería de implantes ortopédicos para gestionar todas las operaciones con y sobre estos mismos. No existen estándares que rijan el cómo se debe hacer la gestión de los implantes digitales ortopédicos, lo que trae consigo que los archivos que contienen la información e imágenes de estos implantes sean específicos de la compañía que los diseñó y únicamente para el software de la misma, imposibilitando que otro software de planificación haga uso de estos.

Las compañías que actualmente desarrollan software para la planificación quirúrgica son privativas, por lo cual no se tiene acceso a ninguna información referente a la manipulación de los implantes digitales. Esto implica que no se puedan utilizar referencias ni aplicar comparaciones a la hora de definir el formato más adecuado para almacenar la información de un implante.

1.3. Organizadores y editores de imágenes

1.3.1. *Studio Line Photo Classic*

Studio Line Photo Classic 3 Plus es un software igualmente poderoso y fácil de usar para la gestión práctica y la edición de colecciones de imágenes amplias para cualquier usuario de fotografía digital.

Studio Line Photo Classic 3 Plus gestiona imágenes de todos los formatos gráficos más comunes en una base de datos segura "Archivo de Imágenes".

Clasifica las imágenes con descripciones textuales de cualquier longitud en los descriptores del sistema o los descriptores propios del usuario, o asignar palabras claves y valoraciones. Después de eso, la localización de imágenes individuales o encontrar imágenes para que coincida con un determinado tema será una tarea trivial.

Studio Line Photo Classic3 Plus viene con una impresionante variedad de herramientas de edición profesional. Con algunos pasos simples que usted puede convertir un conjunto de imágenes en una presentación de diapositivas, publicarlas como una galería web, grabar un CD/DVD o enviarlas a amigos y familiares a través de correo electrónico. También incorporados con amplias funciones de impresión de cualquier formato y tamaño imaginable, o puede imprimir hojas de contactos, tarjetas de felicitación y calendarios. (5) (Ver Anexo 2)

1.3.2.*Zoner Photo Studio 13 PRO Edition*

El Catálogo de filtrado del panel le permite mostrar fotos catalogadas en base a búsquedas de texto completo de la información de su imagen-por ejemplo, clasificación, etiqueta, o la localización GPS, incluyendo el filtrado de varios tipos de información a la vez. También hemos añadido la posibilidad, a menudo solicitado para gestionar automáticamente los archivos asociados, junto con imágenes (por ejemplo JPEG y RAW pares).

- Catálogo de filtrado panel contexto completo de búsqueda y filtrado por las calificaciones, las etiquetas, los datos GPS y mucho más.
- Fácil eliminación automática/mudanza de los archivos relacionados, como JPEG y pares de archivo RAW.
- Rápida selección de archivos es ahora compatible con la selección por calificación o la etiqueta. (6)

Zoner Photo Studio incluye numerosas herramientas para mejorar las fotos y obtener los mejores resultados. La nueva versión trae una gran capacidad solicitada, editar valores de transparencia. Imágenes transparentes ahora se pueden editar y guardar en formatos con valores de transparencia, sin daños, y la transparencia se puede agregar con la nueva herramienta Borrador. Las herramientas

de selección también se han mejorado y ampliado para hacer cambios y collages, más precisa que nunca. (7) (Ver Anexo 3)

1.3.3. *Ashampoo Photo Commander*

Ashampoo Photo Commander 9 es el compañero ideal para toda la gestión, optimización y edición de los procesos relativos a sus fotos digitales. La vista en miniatura en la interfaz de usuario le permite obtener una visión general de su colección de fotos o más bien las fotos en una carpeta. Por supuesto, usted también puede ordenar sus fotos según la fecha y hora en que se tomó la misma ("Información general del calendario") o puede filtrar las fotos según diferentes criterios, por ejemplo, solo los archivos. JPG y no más de dos meses o con la etiqueta un plazo determinado. No solo las imágenes, sino también colecciones de audio y video, se muestran claramente. Las fotos pueden ser agrupadas de manera diferente o compiladas en un álbum virtual para su posterior edición. El procesamiento por lotes le permite seleccionar varias fotos para la edición, por ejemplo, convertirlas en un formato diferente.

La edición y los modos de solución rápida en Ashampoo Photo Commander 9 son claramente diseñados e intuitivos, para que pueda dar rienda suelta a su creatividad y experimentar con efectos y funciones. Es fácil para rotar imágenes, ajustar el alineamiento horizontal, recortar las imágenes, eliminar ojos rojos y mucho más. Con solo un clic en las fotos se puede optimizar o modificar con efectos, por ejemplo, la inversión, pixelización o esbozar los contornos. Ashampoo Photo Commander 9 también le ofrece varias características interesantes a presentar sus fotos. Cree, por ejemplo, una presentación de diapositivas o un álbum HTML para mostrar el resultado de sus fotos a familiares y amigos. Collages, así como tarjetas o calendarios son fáciles de producir en poco tiempo. Además, Ashampoo Photo Commander 9 le permite realizar capturas de pantalla, imágenes de exploración, así como imprimir fotos, enviarlas por correo electrónico o grabarlas en un disco. (8) (Ver Anexo 4)

Características:

- 3D de aceleración de hardware.
- Nuevos y más eficaces medios de comunicación de base de datos.
- Arrastrar y Soltar álbumes virtuales.
- Crear imágenes panorámicas.
- Presentaciones de diapositivas con efecto "Ken Burns".

- Cámara digital y escáner importador.
- Encontrar imágenes duplicadas con nombres diferentes.
- Procesamiento por lote.
- Audio y video.
- Pieles y plugins.

1.4. UML

Se utilizará UML 2.1 como lenguaje de modelado el cual permite especificar, visualizar y documentar los artefactos generados a lo largo del ciclo de desarrollo del software, fundamentalmente en las fases de análisis, diseño e implementación.

UML está basado principalmente sobre los conceptos de la orientación a objetos (OO) y es el más utilizado para lenguajes orientados a objetos como C++, Java y C#. UML no esta vinculado a alguna metodología, lo que ha incidido en su amplio uso para el desarrollo de software.

1.5. Managed Extensibility Framework (MEF)

El uso de MEF simplifica la creación de aplicaciones extensibles. MEF ofrece capacidades de descubrimiento y composición que puede aprovechar para cargar las extensiones de las aplicaciones. MEF presenta una solución sencilla para el problema de ampliación en tiempo de ejecución. MEF proporciona una forma estándar para la aplicación host para exponerse y consumir extensiones externas. Extensiones, que por su naturaleza, se pueden reutilizar entre diferentes aplicaciones. Sin embargo, una extensión podría ser implementada en una manera específica para una aplicación. Las extensiones pueden depender una de otra y el MEF se asegurará de que estén interconectadas en el orden correcto. MEF ofrece un conjunto de métodos de descubrimiento que permite a su aplicación, buscar y cargar las extensiones disponibles, y además permite extensiones con metadatos que facilitaría un mejor filtrado. (9)

1.5.1. *Exchange Modeling Language (XML)*

Es un meta-lenguaje que permite definir lenguajes de marcado adecuados a usos determinados. En la práctica corresponde a un estándar que permite a diferentes aplicaciones interactuar con facilidad a través de La Red. XML fue creado al amparo del World Wide Web Consortium (W3C) organismo que vela por el desarrollo de WWW partiendo de las amplias especificaciones de SGML. Su desarrollo se comenzó en

1996 y la primera versión salió a la luz el 10 de febrero de 1998. La primera definición que apareció fue: Sistema para definir, validar y compartir formatos de documentos en la web. (10)

XML es un lenguaje que permite jerarquizar y estructurar la información y describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo. La información estructurada presenta varios contenidos (texto, imágenes, audio, etc.) y formas: hojas de cálculo, tablas de datos, libretas de direcciones, parámetros de configuración, dibujos técnicos, etc. La forma da alguna indicación de qué papel puede jugar el contenido (por ejemplo, el contenido de una sección encabezada con un significado difiere del contenido de una nota a pie de página, lo que significa algo diferente que el contenido de un pie de foto o el contenido de una tabla de datos). Más o menos todos los documentos tienen la misma estructura. (11)

Debido a las características del XML es posible crear ficheros en este formato que posibiliten el almacenamiento de los datos de un implante digital ortopédico de forma eficiente y escalable, evitando que una modificación en la estructura de un implante inutilice archivos previamente generados.

1.6. Metodología de desarrollo

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. En ingles, Software Development Methodology (SDM) o System Development Lifecycle (SDLC).

La finalidad de una metodología de desarrollo es garantizar la eficacia (p.ej. cumplir los requisitos iniciales) y la eficiencia (p.ej. minimizar las pérdidas de tiempo) en el proceso de generación de software. Los riesgos a afrontar y los controles a establecer varían en función de las diferentes etapas del ciclo de vida de desarrollo. De forma general podríamos encontrar las siguientes fases:

- Definición del proceso de negocio y los requerimientos
- Documentación funcional
- Arquitectura y diseño técnico
- Codificación y ejecución de pruebas unitarias
- Pruebas globales del sistema
- Pruebas de integración

- Implantación
- Formación de usuarios
- Mantenimiento del sistema

Adicionalmente, durante todo el ciclo de vida del proyecto se deberán realizar tareas tales como:

- Gestión de la configuración: identificación de versiones, control de cambios, etc.
- Gestión de la calidad: seguimiento de errores, revisiones del nivel de calidad.
- Revisión de las premisas iniciales: revisión de los requerimientos y de los diseños.
- Gestión del entorno de desarrollo: herramientas de desarrollo, librerías, ficheros, gestión de datos (p.ej. para pruebas) (12)

1.6.1. *Open Unified Process*

OpenUP/Basic es un Framework de procesos de desarrollo de software de código abierto. Es un proceso modelo y extensible, dirigido a gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.

Este proceso de desarrollo unificado está basado en RUP[1], desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad. Plantea que sólo se deben usar los procesos que sean necesarios, que no se deben usar muchos artefactos, además debe acoplarse a las necesidades del usuario permitiendo ser modificado y extendido.

OpenUP está caracterizado por cuatro principios básicos, los cuales son:

- Colaboración para unificar intereses y compartir conocimientos.
- Equilibrio de prioridades competentes a maximizar el valor de los involucrados con el resultado del proyecto.
- Enfoque en la articulación de la arquitectura.

[1] Rational Unified Process.

- Desarrollo continuo para obtener retroalimentación y realizar las mejoras respectivas. (13).

1.6.2.*Rational Unified Process (RUP)*

RUP es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable. (14) Es un proceso de desarrollo de software que contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Más que un simple proceso, el proceso de desarrollo de software es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de actitud y diferentes tamaños de proyecto. (15)

Las características principales de RUP son:

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan, es una facilidad que el software debe proveer a sus usuarios. A partir de aquí los casos de uso guían el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.
- **Centrado en la arquitectura**: La arquitectura involucra los elementos más significativos del sistema. Surge de las necesidades de la empresa y se refleja en los casos de uso. También se ve influida por otros factores como las plataformas de software, los sistemas operativos, protocolos y requerimientos no funcionales. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de Lenguaje Unificado de Modelado (UML).
- **Iterativo e incremental:** RUP divide el proceso en cuatro fases, las cuales se desarrollan en iteraciones que incluyen las actividades principales básicas de cualquier proceso de desarrollo. Las iteraciones hacen referencia a pasos en los flujos de trabajo y los incrementos, al crecimiento del producto. Las iteraciones están controladas y se ejecutan de una forma planificada. Se selecciona RUP como metodología de desarrollo porque provee un entorno de proceso de desarrollo configurable, basado en estándares, permite tener claro y accesible el proceso de desarrollo que se sigue, además de ser configurado según las necesidades de la organización y del proyecto.

Utiliza mejores prácticas probadas en la industria y provee a cada participante con la parte del proceso que le compete directamente, filtrando el resto. Dada las características de esta metodología y la serie de bondades que brinda al aplicarla, hacen de ella una buena elección. Tomando en cuenta también que esta misma ha sido adoptada por el CESIM como metodología de desarrollo por excelencia.

1.7. Herramientas

Las herramientas de desarrollo son aplicaciones que influyen de manera significativa en el desarrollo de un programa. A continuación se describen las principales características de las herramientas Enterprise Architect, Visual Studio 2010, Subversion (SVN) y Tortoise SVN; las cuales fueron seleccionadas para dar cumplimiento al objetivo general.

1.7.1. *Enterprise Architect 7.5*

Enterprise Architect (EA) combina el poder de la última especificación UML 2.1 con alto rendimiento, interfaz intuitiva, para traer modelado avanzado al escritorio, y para el equipo completo de desarrollo e implementación. Con un gran conjunto de características y un valor sin igual para el dinero, EA puede equipar a su equipo entero, incluyendo analistas, evaluadores, administradores de proyectos, personal del control de calidad, equipo de desarrollo y más, por una fracción del costo de algunos productos competitivos.

Enterprise Architect es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multiusuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad. El manual de usuario está disponible en línea. El UML provee beneficios significativos para ayudar a construir modelos de sistemas de software rigurosos y donde es posible mantener la trazabilidad de manera consistente. Enterprise Architect soporta este proceso en un ambiente fácil de usar, rápido y flexible.

Enterprise Architect provee trazabilidad completa desde el análisis de requerimientos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue. Combinados con la ubicación de recursos y tareas incorporados, los equipos de Administradores de Proyectos y Calidad están equipados con la información que ellos necesitan para ayudarles a entregar proyectos en tiempo. Las bases de

Enterprise Architect están construidas sobre la especificación de UML 2.0. Usa perfiles UML para extender el dominio de modelado, mientras que la Validación del Modelo asegura integridad. Combina Procesos de Negocio, Información y Flujos de trabajo en un modelo usando nuestras extensiones gratuitas para BPMN y el perfil Eriksson-Penker. (16)

1.7.2.*Microsoft Visual Studio 2010*

Como entorno de desarrollo integrado (IDE) se ha seleccionado al Microsoft Visual Studio. Este mismo Visual Studio es un completo conjunto de herramientas de desarrollo para construir aplicaciones Web ASP.NET, XML Web Services, las aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Además, estos lenguajes que utilizan la funcionalidad de .NET Framework, que proporciona acceso a las tecnologías claves para simplificar el desarrollo de aplicaciones Web ASP y Servicio Web XML.

Este IDE hace uso del .NET Framework 4.0 que es un componente integral de Windows que admite la creación y funcionamiento de la próxima generación de aplicaciones y servicios Web. Los principales componentes de .NET Framework 4.0 son: el Common Language Runtime(CLR) y .NET Framework Class Library, que incluye ADO.NET, ASP.NET, Windows Forms y Windows Presentation Foundation(WPF). El .NET Framework proporciona un entorno de ejecución administrado, desarrollo e implementación simplificados, y la integración con una amplia variedad de lenguajes de programación. (17)

1.7.3.*Subversion (SVN) y Tortoise SVN 1.6.6*

Es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se lo conoce también como SVN por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Ventajas:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.

- La creación de ramas y etiquetas es una operación más eficiente; Tiene costo de complejidad constante (O(1)) y no lineal (O(n)) como en CVS.
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Puede ser servido, mediante Apache, sobre Web DAV/DeltaV. Esto permite que clientes Web DAV utilicen Subversion en forma transparente.
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.). (18)

Tortoise SVN es un software para la revisión y control de versiones/software de control de código fuente para Windows. Ya que no está integrado a ningún IDE específico que se puede utilizar con cualquier herramienta de desarrollo. Tortoise SVN es de uso libre. (19)

1.8. Patrones de diseño

1.8.1. *Patrón Inyección de Dependencias e Inversión de Control.*

Inversion of Control (IoC). Es una técnica que invierte el flujo tradicional. Lo tradicional es que el código que implementes llame a las librerías; la inversión de control ocurre cuando son las librerías las que llaman a tu código. En Spring, la inversión de control consiste en ceder el control a una entidad externa a la aplicación, llamada "*Contenedor*", que se encargará de gestionar las instancias (así como sus creaciones y destrucciones).

Dependency Injection (DI). Es un término comúnmente confundido con el anterior. En un escenario en el que utilizamos IoC, delegamos en una entidad "*Contenedor*" no solo la gestión de las instancias, sino la inyección de las sub-partes (dependencias). Si una venta se compone de un cliente y un producto, al instanciar un objeto venta la DI le inyecta directamente su cliente y su producto específicos. (20)

1.8.2.*Patrones GRASP*

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza, que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (21)

1.8.3.1 El Patrón Experto

EL patrón experto asigna una responsabilidad al experto en información, es decir, la clase que tiene la información necesaria para cumplir con la responsabilidad. El problema que resuelve el patrón experto está referido al principio más básico mediante el cual las responsabilidades son asignadas en el diseño orientado a objetos.

1.8.3.2 El Patrón Creador

Este patrón asigna a la clase **B** la responsabilidad de crear una instancia de la clase **A** si alguna de las siguientes premisas es cierta:

- **B** agrega objetos de **A**.
- **B** contiene objetos de **A**.
- **B** registra instancias de objetos de **A**.
- **B** usa muchos objetos de **A**.
- **B** tiene la data inicial que será pasada a **A** cuando es creada (así **B** es un experto con respecto a la creación de **A**).

B es el creador de los objetos de **A**. Si más de una opción aplica, se prefiere una clase de **B** que agrega o contiene a la clase **A**. El uso de este patrón presenta un problema interesante que consiste en elegir quién debe ser el responsable de crear una nueva instancia de alguna clase. La creación de objetos es una de las actividades más comunes en un sistema orientado a objetos, Consecuentemente, es muy útil tener un principio general para la asignación de la creación de responsabilidades. Asignados de una manera correcta, el diseño soporta un bajo acoplamiento, incrementa la claridad, la encapsulación y es posible rehusarlo. (22)

1.8.3.3 El Patrón Polimorfismo

Este patrón se utiliza cuando las alternativas o comportamientos relacionados varían según la clase. Asigna la responsabilidad para el comportamiento, utilizando operaciones polimórficas a los tipos para los que varía el comportamiento. Tiene como ventajas que se añaden fácilmente las extensiones necesarias para nuevas variaciones y las nuevas implementaciones se pueden introducir sin afectar a los clientes.

En este capítulo se evaluaron las tendencias actuales en la construcción de librerías de software identificando la estructura más adecuada para dar solución al problema planteado. Se seleccionó el formato de archivo XML para el almacenamiento de sus datos dada la flexibilidad de esta estructura para guardar distintos tipos de datos, además del empleo de comandos gráficos para su representación visual.

Para el desarrollo del componente se valoraron tecnologías y herramientas existentes seleccionándose al C# 4.0, Framework .NET 4.0, Enterprise Architect 7.5, Visual Studio 2010 y para el manejo y control de versiones el Tortoise Subversion.

CAPÍTULO 2. **Características del sistema**

En este capítulo se presenta la propuesta de solución del sistema para la situación problémica. Se muestran los procesos del negocio mediante el modelo de dominio. Se evidencian las características y funcionalidades que tendrá el sistema, a partir de los requerimientos funcionales y no funcionales del componente de software a desarrollar. Se presenta el diagrama de casos de uso del sistema con las correspondientes especificaciones de los casos de uso asociados al componente.

2.1. Breve descripción del sistema

Actualmente la planificación quirúrgica ortopédica se lleva a cabo utilizando implantes digitales, para los cuales no existen estándares o componentes individuales que permiten su gestión y visualización. Teniendo en cuenta esto, se hace necesario desarrollar un componente que permita su manipulación, así como una herramienta que permita crearlos y modificarlos. Ya que además, tampoco existe un editor específico para implantes digitales, ni libre ni privativo que esté a la venta o se tenga conocimiento de él.

2.2. Modelo de dominio

En la primera fase de desarrollo de RUP, este define la realización del modelo de negocio, cuyo objetivo está en comprender el entorno del cliente y detectar las mejoras en los procesos de la organización. El modelo de dominio es un subconjunto del modelo de negocio, utilizado cuando no es posible identificar claramente los procesos del negocio, en este se capturan los tipos de objetos más importantes del sistema. Los objetos del dominio identifican conceptos o eventos que ocurren en el entorno del trabajo del sistema. El modelo de dominio se describe utilizando el lenguaje de modelado UML mediante diagrama de clases.

Teniendo en cuenta que los procesos de negocio no están claramente definidos, se describe el negocio a través del modelo de dominio. (Ver Figura 2)

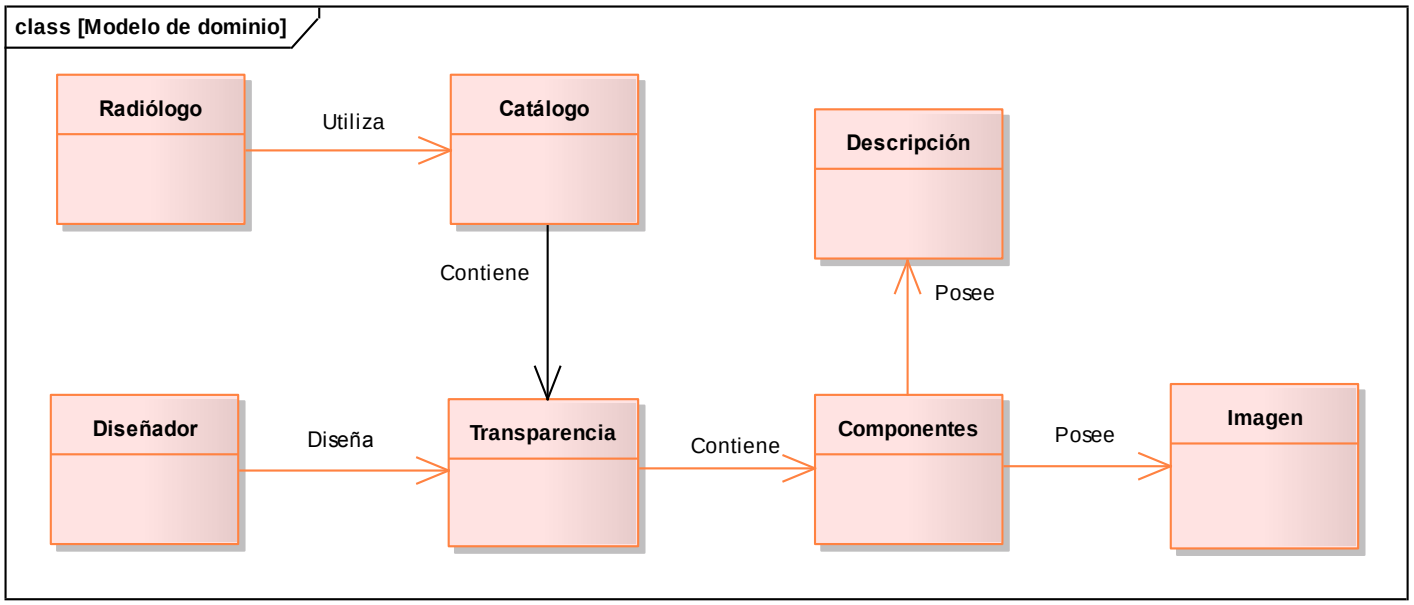


Figura 2. Modelo de dominio.

A continuación, para lograr un mayor entendimiento del sistema, se brinda una descripción de los principales elementos del modelo de dominio facilitando de esta manera la comprensión del contexto que se describe.

2.2.1. *Elementos del dominio*

- Catálogo: Engloba la organización y manipulación de las transparencias digitalizadas.
- Componentes: Concepto que engloba las particularidades de las transparencias digitales.
- Descripción: Representa las características de los componentes de las transparencias digitales.
- Diseñador: Representa las operaciones de diseño de una transparencia digital.
- Radiólogo: Concepto que engloba el uso de las transparencias digitales.
- Imagen: Identifica la representación visual de las transparencias digitales.
- Transparencia: Representa las características de un implante ortopédico.

2.3. Especificación de los requisitos de software

Los requisitos funcionales y no funcionales muestran las capacidades o condiciones que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener, los cuales en la fase de construcción deben ser posibles de probar o verificar.

2.3.1.*Requisitos funcionales.*

Nº	Funcionalidad	Descripción	Complejidad
RF 1 Cargar Implante	Cargar implante.	Un sistema externo provee criterios de búsqueda para los cuales el sistema localiza el archivo específico y lo carga	Baja
Paquete RF 2 Dibujar Implante			
RF 2.1	Mostrar la representación visual de un implante.	Dado un índice de sub-imagen el sistema dibuja en un contexto de dibujo dicha imagen bajo ciertos parámetros de posición y rotación.	Media
RF 2.2	Rotar implante dado sobre su punto medio.	Dado un ángulo crear una matriz de rotación para modificar la forma de visualización de la sub-imagen del implante	Media
RF 2.3	Trasladar implante sobre una imagen.	Dado un punto mover y redibujar el implante sobre dicho punto.	Baja
Paquete RF 3 Gestionar Diseño			
RF 3.1	Crear un diseño de implante.	Permitir al diseñador crear nuevos diseños, almacenar la	Baja

		información de estos en un archivo o suprimirlos de la edición.	
RF 3.2	Almacenar un diseño de implante.	Permitir el almacenamiento del diseño de un implante en un formato de archivo flexible.	Baja
RF 3.3	Suprimir un diseño de implante.	Permitir la eliminación de un diseño.	Baja
RF 3.4	Editar la información referente a un implante.	Permitir al diseñador introducir la información textual referente al implante.	Baja
RF 3.5	Editar la representación visual del implante.	Permitir al diseñador hacer uso de los comandos gráficos para diseñar el implante.	Alta
RF 3.6	Pre-visualizar diseño	Permitir la pre-visualización del diseño.	Media

2.3.2.*Requisitos no funcionales*

Apariencia o interfaz externa.

RNIU 1: El componente de software contará con una interfaz amigable y similar a las aplicaciones del sistema operativo Windows para brindar facilidades de uso.

Funcionamiento

RNFO 1: Se debe disponer para el uso del componente, del sistema operativo Windows o superior.
RNFO 2: Se debe instalar .Net Framework v4.0.
RNFO 3: Pentium V 3.0 GHz o superior.
RNFO 4: Disco rígido 20 GB.
RNFO 5: Memoria de video de 512 Mb o superior.

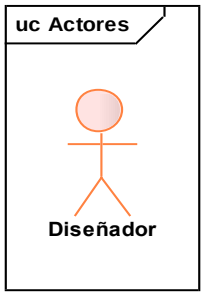
RNFO 6: Memoria principal DRR 1Gb.

Restricciones en el diseño e implementación.

RNDI1: El análisis y diseño del componente será basado en la metodología RUP con el uso del lenguaje de modelado UML.
RNDI2: Se usará como herramienta CASE, Enterprise Architect para el modelado de los artefactos que se generan en cada uno de los flujos de trabajo.
RNDI 3: Se utilizará como IDE de desarrollo Visual Studio 2010.
RNDI 4: Se usará como lenguaje de programación C#.

2.4. Definición de los casos de uso del sistema.

2.4.1. *Actor del sistema*

Actor	Descripción
Diseñador 	Es la persona que interactúa con el sistema para editar los implantes digitales ortopédicos con el fin de que puedan ser empleados en un sistema de planificación quirúrgica.
Planificador 	Es el sistema que hace uso de las funcionalidades del componente para la visualización de los implantes digitales ortopédicos.

A continuación se muestran los diagramas de casos de uso del sistema.

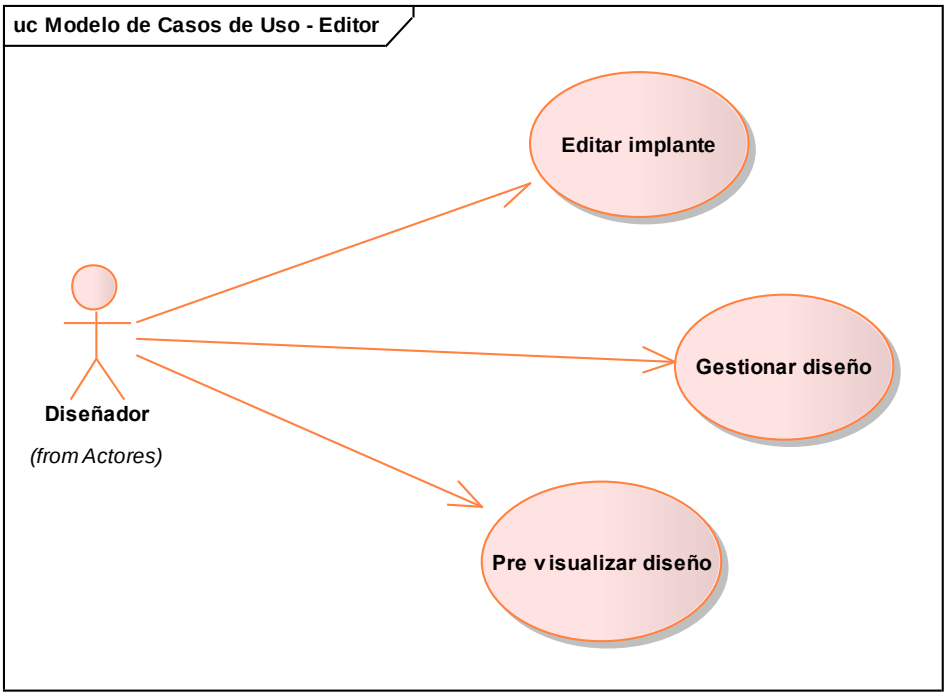


Figura 3. Diagrama de Casos de Uso del Editor

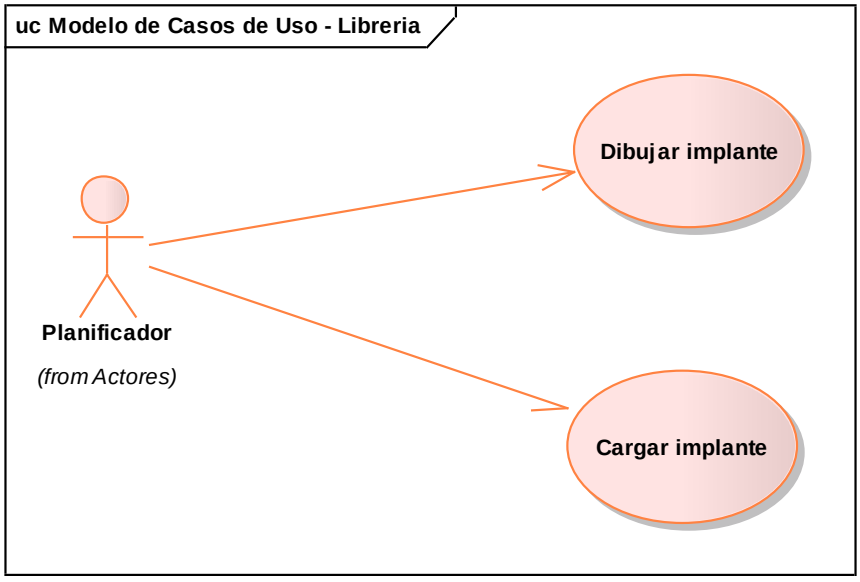


Figura 4. Diagrama de Casos de Uso de la Librería

2.4.2.*Casos de uso por ciclo*

- Primer ciclo:

Cód.	Nombre del caso de uso	Paquete	Justificación
CU 1	Cargar implante	Caso de uso del sistema (Crítico)	Estos casos de uso describen los procesos a automatizar específicos a la manipulación de los implantes digitales ortopédicos.
CU 2	Dibujar implante	Caso de uso del sistema (Crítico)	

- Segundo ciclo:

Cód.	Nombre del caso de uso	Paquete	Justificación
CU 3	Gestionar diseño	Caso de uso del sistema (Crítico)	Estos casos de uso describen los procesos a automatizar específicos al diseño de los implantes digitales ortopédicos.
CU 4	Editar implante	Caso de uso del sistema (Crítico)	
CU 5	Pre-visualizar diseño	Caso de uso del sistema (secundario)	

2.4.3.*Casos de uso expandidos*

Los casos de uso expandidos son muy útiles para alcanzar un conocimiento más profundo y detallado de los procesos y de los requerimientos. Constituyen un documento narrativo que describe la secuencia de eventos de un actor que utiliza el sistema para completar el proceso. Las tablas extendidas se encuentran en el Anexo 5.

A continuación se muestran los casos de uso simplificados permitiendo tener una noción básica del sistema.

CU # 1. Cargar Implante	
Objetivo	Cargar un implante dado criterios de búsqueda.
Actores	Planificador
Resumen	Permitir que el planificador suministre información para realizar una búsqueda y localizar el archivo en cuestión, y a su vez cargar la información de este.
Complejidad	Baja
Prioridad	Crítico
Referencias	RF 1.1

CU # 2. Dibujar Implante	
Objetivo	Dibujar un implante en un contexto de imagen.
Actores	Planificador
Resumen	Un conjunto de acciones y transformaciones necesarias para que el planificador pueda visualizar el implante.
Complejidad	Media
Prioridad	Crítico
Referencias	RF 2.1, RF 2.2, RF 2.3

CU # 3. Gestionar diseño	
Objetivo	Diseñar un implante digital ortopédico.
Actores	Diseñador
Resumen	El planificador le brinda criterios de búsqueda dado los cuales el sistema encuentra el archivo correspondiente al implante indicado y carga la información

	correspondiente al mismo.
Complejidad	Baja
Prioridad	Crítico
Referencias	RF 3.1

CU # 4. Editar contenido del implante	
Objetivo	Editar el contenido del implante
Actores	Diseñador
Resumen	Conjunto de operaciones necesarias para editar la información del implante y diseñar su representación visual.
Complejidad	Alta
Prioridad	Crítico
Referencias	RF 3.2, RF 3.3

CU # 5. Pre-visualizar diseño	
Objetivo	Pre-visualizar un implante en edición
Actores	Diseñador
Resumen	Un conjunto de operaciones necesarias para pre-visualizar el diseño de un implante.
Complejidad	Media
Prioridad	Secundario
Referencias	RF 3.4

En el presente capítulo fueron elaborados y descritos brevemente algunos de los artefactos propuestos por RUP para el desarrollo de software, tales como el modelo de dominio, definición de los requisitos funcionales y no funcionales, los actores, casos de uso, diagrama de casos de uso del sistema y la descripción textual de los mismos.

CAPÍTULO 3. Análisis y diseño

Este capítulo describe cómo implementar el componente, a través del diseño, enfocado a cómo el sistema cumple con sus objetivos teniendo en cuenta los requisitos funcionales y no funcionales. Se realizan los diagramas de clases y los diagramas de interacción según los casos de uso definidos y se explica además la arquitectura utilizada, así como el empleo de los patrones de diseño.

3.1. Arquitectura del sistema

3.1.1. *Estilo arquitectónico utilizado*

Para el desarrollo del componente se hace uso del patrón arquitectónico Modelo Vista Controlador. Este patrón, se basa en la separación de las tres capas del diseño de las aplicaciones: el modelo de datos, la presentación de los mismos y las acciones de los usuarios. El componente hace uso intensivo de las interfaces visuales que requieren de una actualización constante de las propiedades que se modifican. Realiza un diseño que desacopla la vista del modelo, con la finalidad de mejorar la reusabilidad, de esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. A continuación se muestra este patrón de arquitectura.

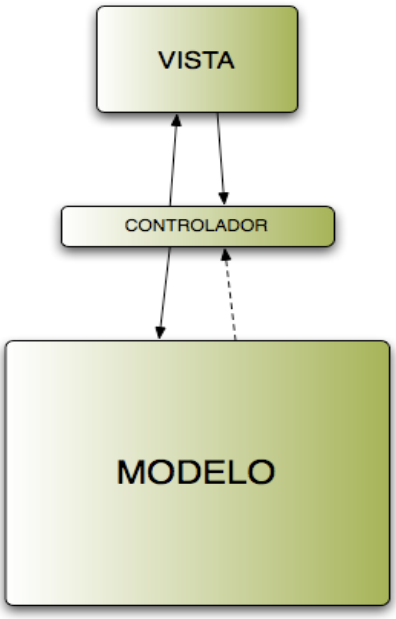


Figura 5. Modelo - Vista - Controlador.

3.1.2.*Patrones utilizados*

1.1.2.1 Patrón Model View – ViewModel

Problema: El uso de WPF brinda posibilidades de optimizar el modo de interacción entre la vista y el modelo, para hacerlo más eficiente.

Solución: Para esto se aplica el patrón MVVM el cual permite, dada su estructura de enlace de datos, que se actualice la vista sin necesidad de escribir código específicamente para dicho propósito.

Aplicación: Este patrón se utiliza para conectar la interfaz de usuario principal del editor, con las clases que representan el modelo de datos y además posee las funcionalidades necesarias para su manipulación.

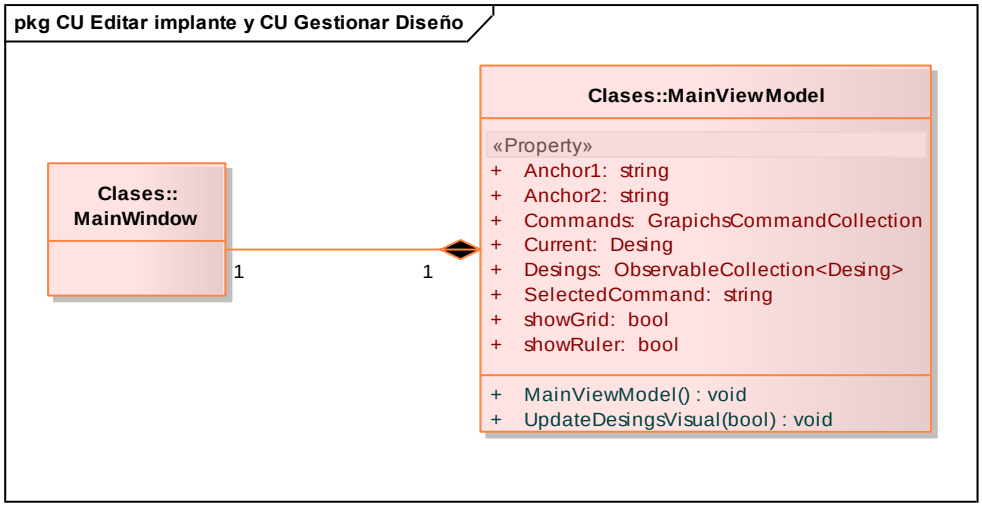


Figura 6. Patrón MVVM. UI Editor.

Aplicación: Este patrón se utiliza para conectar la interfaz de usuario del pre-visualizador con el diseño de implante ortopédico seleccionado, permitiendo además a la interfaz de usuario mostrar los datos referentes al implante así como su representación visual, posee las funcionalidades necesarias para la rotación y traslación del implante digital ortopédico.

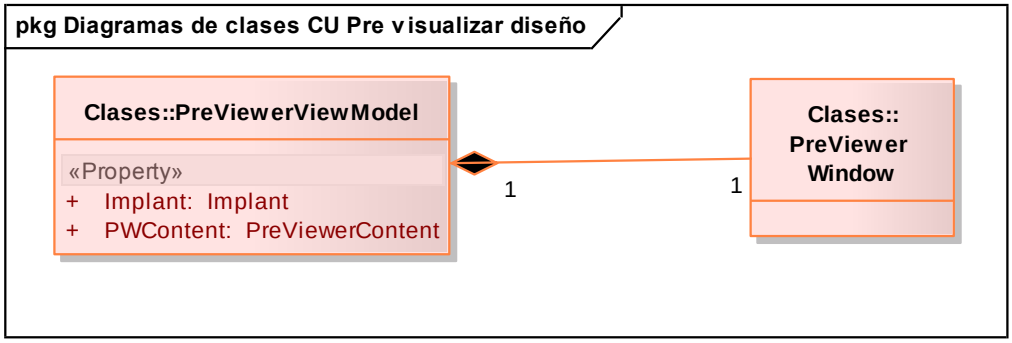


Figura 7. Patrón MVVM. UI Previsualizar

1.1.2.2 Patrón Experto

Problema: Las responsabilidades deben ser acordes a la información que posee cada clase.

Solución: Para esto se aplica el patrón GRASP, Experto.

Aplicación: La clase Implant es la experta en la información, ya que es la que contiene información del implante ortopédico, permite almacenarlo en un fichero o cargar uno, previamente almacenado.

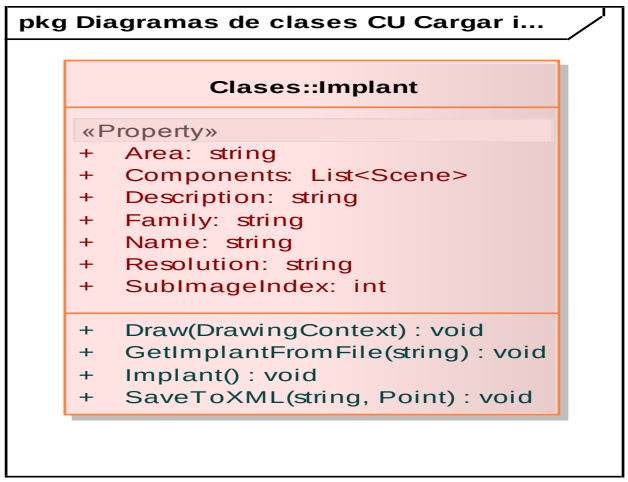


Figura 8. Patrón Experto. Clase Implant.

Aplicación: La clase DrawingArea es la experta en la información, ya que es la que contiene las funcionalidades del diseño implante ortopédico que permiten editarlo a través de la adición o supresión de comandos gráficos y la modificación de los valores de estos.

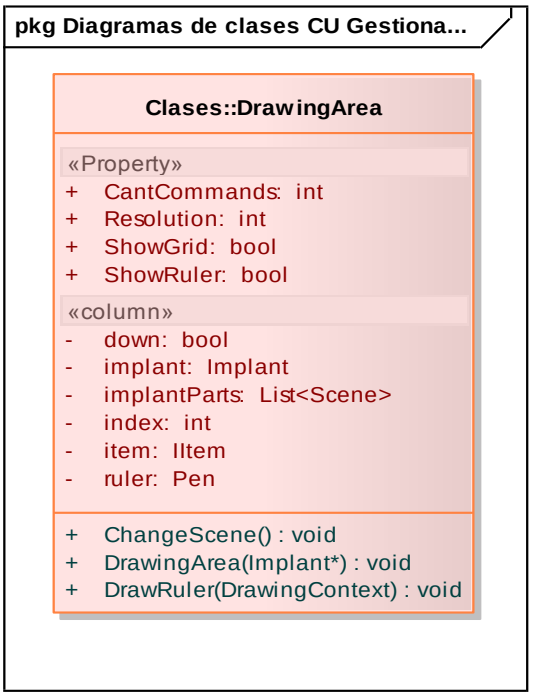


Figura 9. Patrón Experto. Clase DrawingArea.

1.1.2.3 Patrón Creador

Problema: El uso de WPF brinda posibilidades de optimizar el modo de interacción entre la vista y el modelo, para hacerlo más eficiente.

Solución: Para esto se aplica el patrón MVVM el cual permite, dada su estructura de enlace de datos, que se actualice la vista sin necesidad de escribir código específicamente para dicho propósito.

Aplicación: La clase Design es la encargada de crear la clase Implant, ya que es la que posee toda la información necesaria.

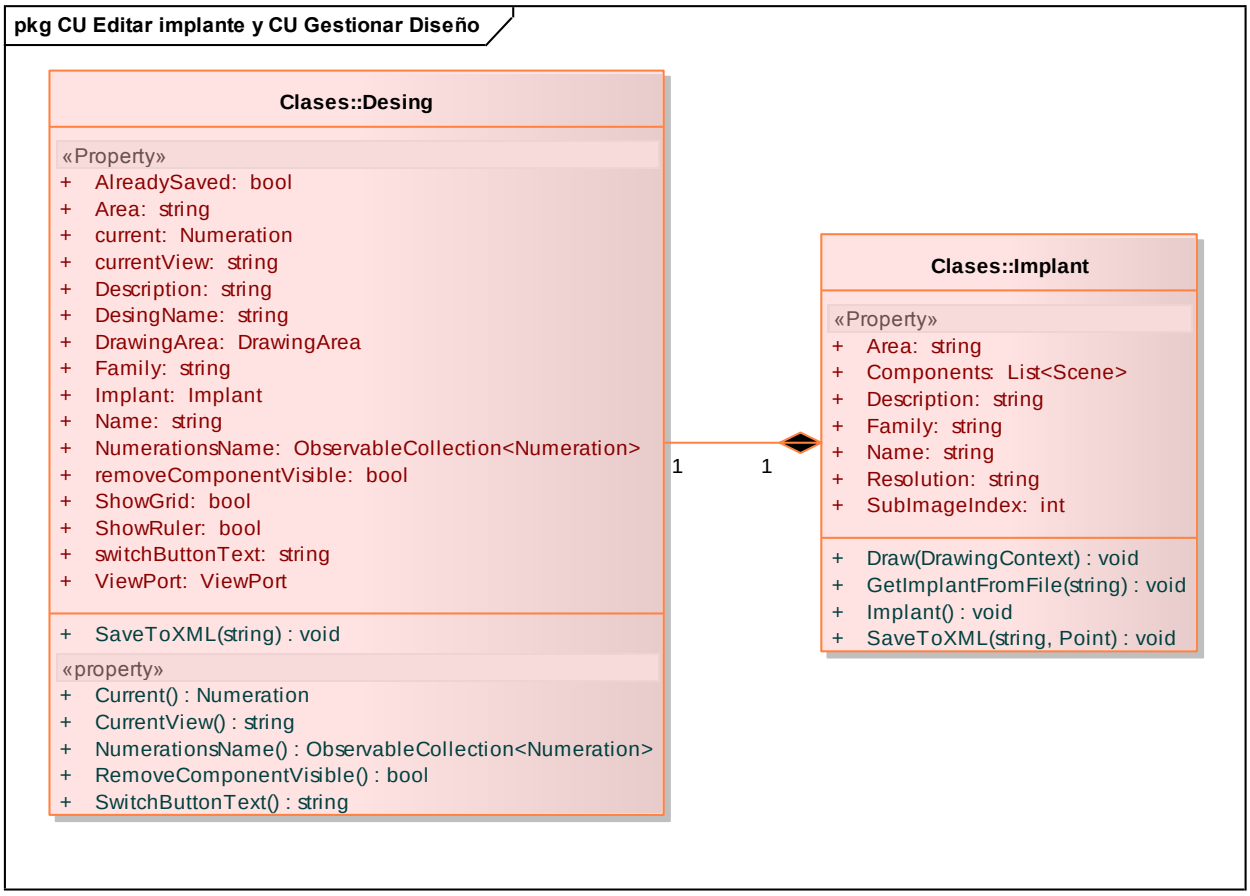


Figura 10. Patrón Creador.

1.1.2.4 Patrón Polimorfismo.

Problema: Cómo manejar diferentes comportamientos en dependencia del tipo de objeto.

Solución: Para dar solución a esto, se aplica el patrón GRASP, Polimorfismo.

Aplicación: La interfaz IDrawableCommand define una serie de métodos que luego son redefinidos por cada uno de los comandos gráficos que se implementan, permitiendo así una aplicación extensible.

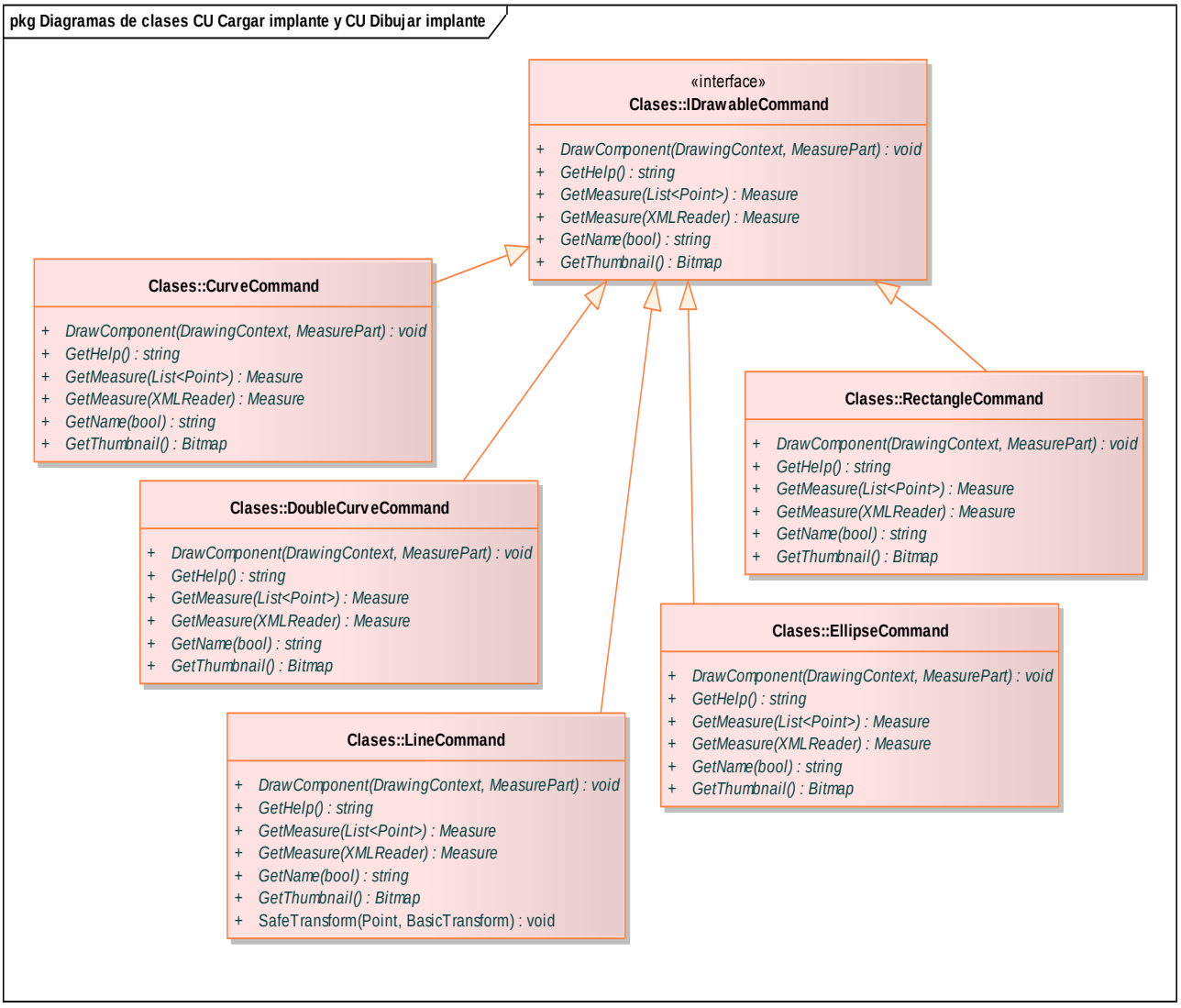


Figura 11. Patrón Polimorfismo. Interface IDrawableCommand

Aplicación: La clase MeasurePart perteneciente a la librería Viewport, define una serie de métodos y propiedades que permiten dibujar y manipular una geometría de 2 dimensiones dado ciertos parámetros que luego son redefinidos por cada una de las geometrías especificas del editor, que se implementan, permitiendo así una aplicación extensible.

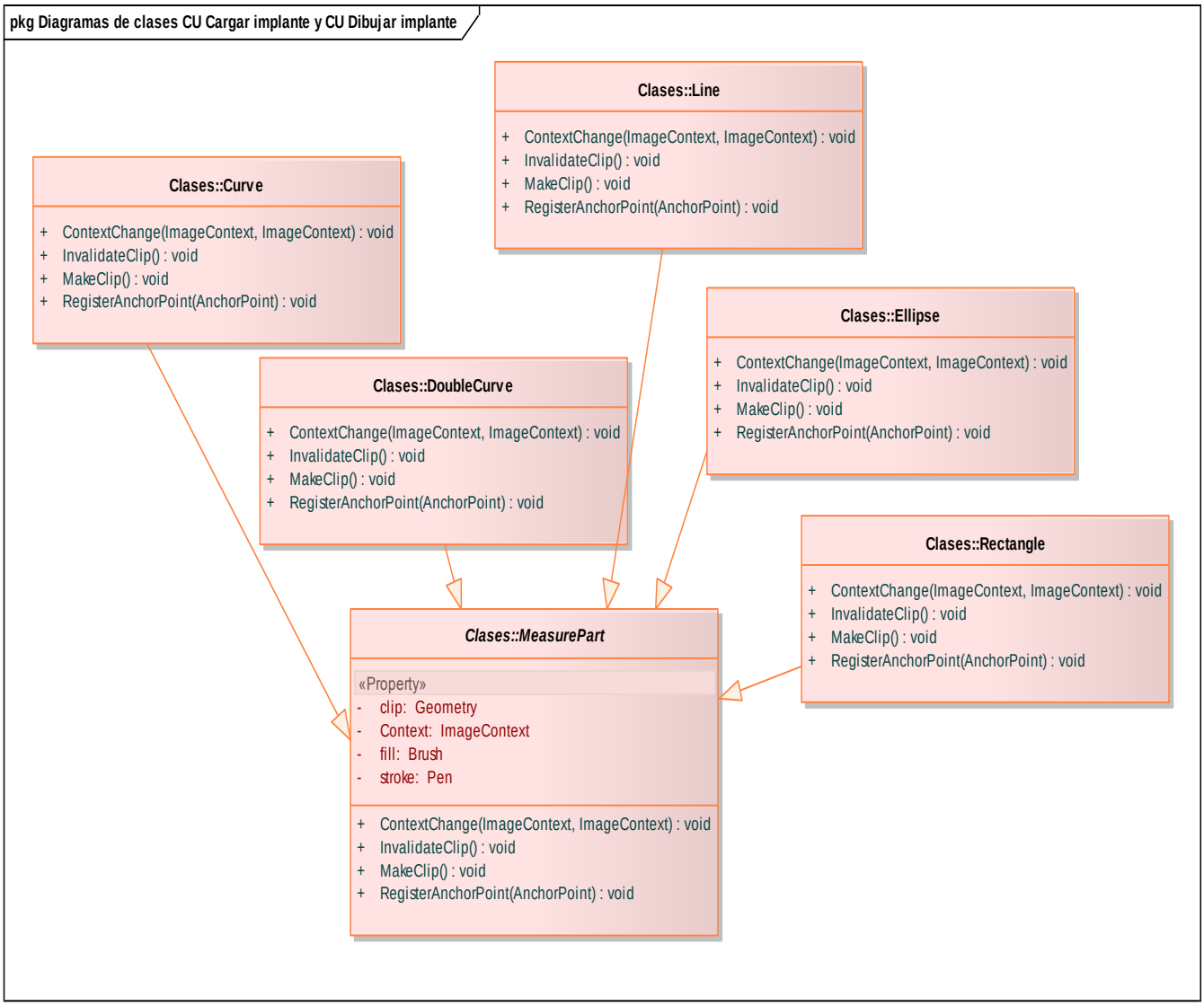


Figura 12. Patrón Polimorfísmo. Clase MeasurePart.

1.1.2.5 Patrón inyección de dependencias.

Problema: Cómo utilizar comandos gráficos permitiendo la mayor flexibilidad posible en su creación.

Solución: Para dar solución a esto se aplica el patrón de inyección de dependencias.

Aplicación: Se utiliza Managed Extensibility Framework para el manejo de las dependencias a utilizar por el sistema, mediante la implementación de la interfaz IImplantsContract.

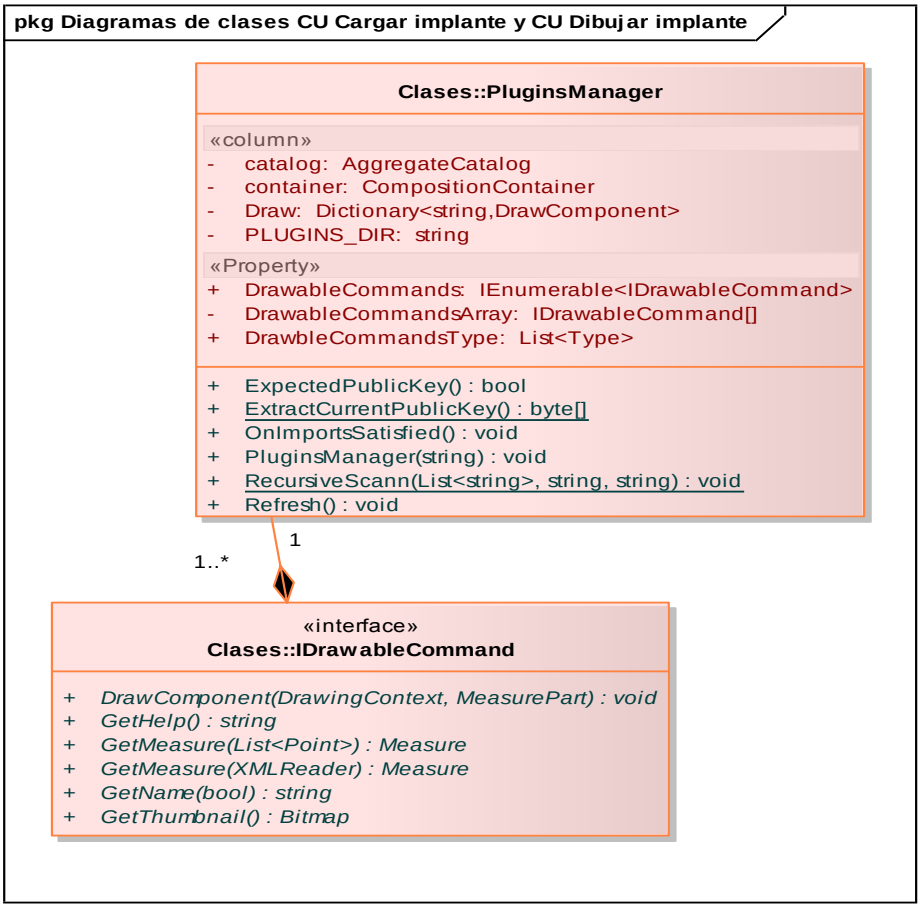


Figura 13. Patrón Inyección de dependencias.

1.1.2.6 Patrón bajo acoplamiento y alta cohesión.

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clase.

3.2. Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que captura la estructura lógica del sistema y describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos, más que cómo se hace algo. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los

sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia y de uso. Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces de la aplicación. El diseño de la aplicación por casos de uso se puede apreciar en el Anexo 6.

3.3. Diagramas de secuencia.

Un diagrama de secuencia es una forma de diagrama de interacción, que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo, representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican, con qué otros objetos y qué mensajes disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos complejos. (23)

La figura 24 (Ver Anexo 7) muestra la interacción entre las clases que componen el caso de uso Editar implante, específicamente el escenario editar información referente al implante. Se evidencia como al introducir los datos estos se validan y el MainViewModel modifica esta información en la estructura de datos subsecuente encargada de almacenar y manipular dichos datos.

La figura 25 (Ver Anexo 8) muestra la interacción entre las clases que componen el caso de uso Editar implante, específicamente el escenario editar representación visual. Se evidencia el rol de mediador del MainViewModel entre la vista y la controladora, permitiendo a comandos y dibujos realizarse sin necesidad de código extra en estas, para propósitos específicos de actualización.

La figura 26 (Ver Anexo 9) muestra la interacción entre las clases que componen el caso de uso Gestionar diseño, específicamente el escenario crear diseño. Se selecciona la opción crear con la cual se lanza el comando CreateDesign, el cual crea un objeto de tipo Design y lo agrega a la colección existente en el MainViewModel.

La figura 27 (Ver Anexo 10) muestra la interacción entre las clases que componen el caso de uso Gestionar diseño, específicamente el escenario eliminar diseño. Se selecciona la opción almacenar, con la cual se lanza el comando DeleteDesign, este procede a eliminar el diseño de la lista de diseños actuales.

La figura 28 (Ver Anexo 11) muestra la interacción entre las clases que componen el caso de uso Gestionar diseño, específicamente el escenario almacenar diseño. Se selecciona la opción almacenar con la cual se lanza el comando SaveDesign, este le muestra al actor un diálogo en el cual se especificará el nombre y lugar de almacenamiento del implante, se crea un archivo de formato XML con dicho contenido e informa de que el diseño acaba de ser guardado.

La figura 29 (Ver Anexo 12) muestra la interacción entre las clases que componen el caso de uso Pre-visualizar diseño, el cual posee un solo escenario. Se selecciona la opción pre-visualizar con la cual se lanza el comando PreView, le muestra al actor una ventana con la información del implante actualmente en edición y una pre-visualización del mismo, con funcionalidades de traslación y rotación.

La figura 30 (Ver Anexo 13) muestra la interacción entre las clases que componen el caso de uso Cargar implante, el cual posee un solo escenario. Se procede a cargar un implante digital ortopédico dado una dirección física. Al ser cargado el implante, este proveerá información respecto al mismo y cuya representación visual estará manipulada por un VisualImplant que el propio implante creará.

La figura 31 (Ver Anexo 14) muestra la interacción entre las clases que componen el caso de uso Dibujar implante, el cual posee un solo escenario. El implante digital ortopédico, es representado visualmente por la clase VisualImplant, la cual posee los funcionalidades necesarias para trasladar y rotar la imagen que identifica al implante digital ortopédico.

En este capítulo quedó establecido el estilo arquitectónico utilizado para el desarrollo del componente; además se abordó la utilización de los patrones de diseño. Las clases del diseño se representaron a través de los diagramas de clases del diseño, y la interacción de las mismas a partir de los diagramas de secuencias.

CAPÍTULO 4. Implementación del sistema

En este capítulo se describe cómo fue implementado el sistema. Se presenta el diagrama de componentes, que muestra la distribución física de los mismos para la implementación. Se describen los fragmentos relevantes del código.

4.1. Diagrama de componentes

Los Diagramas de Componentes ilustran las piezas del software, controladores embebidos, etc. que conformarán un sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase – usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como eventualmente, un componente puede comprender una gran porción de un sistema. (24)

En la figura 14 se puede apreciar el diagrama de componentes del sistema, en el cual se muestran las relaciones existentes entre ellos. EL componente TraumaView Implant Editor se relaciona con 2 librerías; Viewport y TraumaView Implant Library, siendo esta última la responsable de la manipulación de los implantes digitales ortopédicos, esta librería además hace referencia a un conjunto de librerías que contienen los comandos gráficos necesarios para la visualización y manipulación de los implantes digitales ortopédicos.

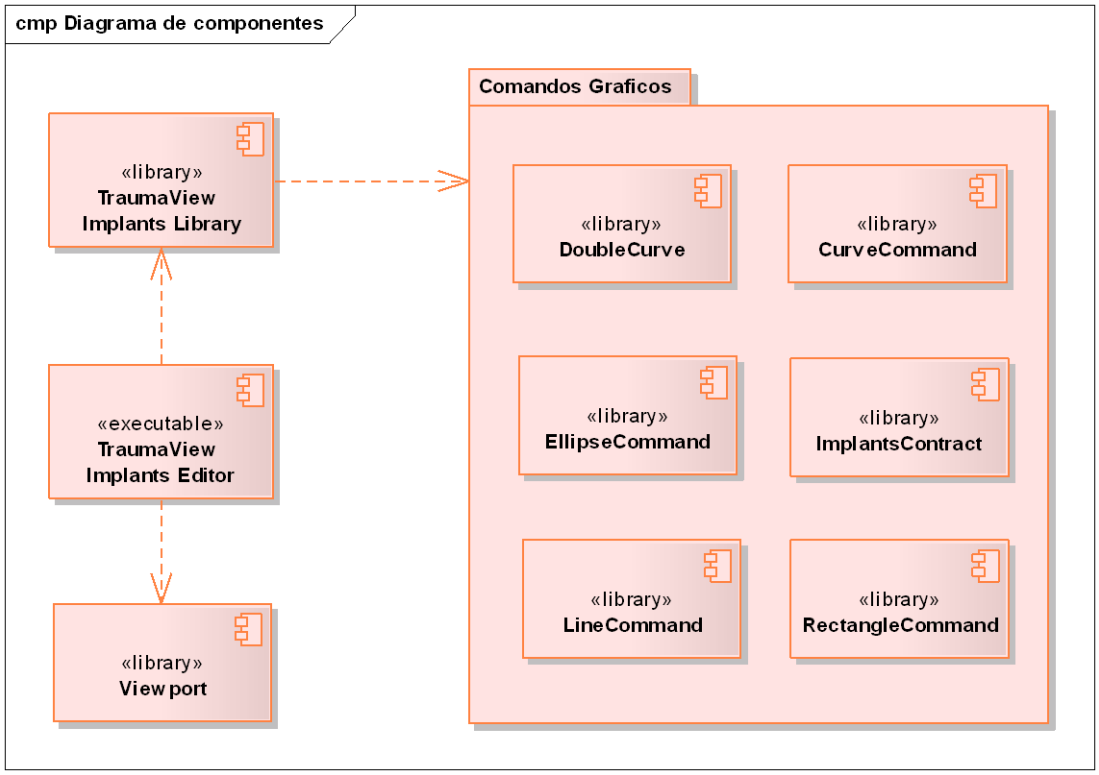


Figura 14. Diagrama de componentes.

4.2. Fragmentos de código

En esta sección se exponen los fragmentos de código críticos en el desarrollo del componente, haciendo énfasis en las técnicas utilizadas almacenar y cargar un implante en archivos XML, así como, cargar las extensiones de la aplicación referente a los comandos gráficos y herramientas utilizadas para diseñar el implante y la creación de estos.

4.2.1. *Guardar diseño en archivo.*

Este fragmento de código exporta el contenido de la clase implante en un archivo XML. El archivo está dividido en secciones las cuales permitirán concentrar y organizar la información del implante. Primero se adicionan los datos referente al implante, como son el nombre, área, familia, etc., luego se identifican cuantas numeraciones posee el implante y se procede a guardar el contenido de cada una ellas en el archivo por secciones individuales para diferenciarlas.

Método SaveToXML (Pseudocódigo)
1. Inicio 2. Crear documentoXML 3. Escribir Descripción_del_implante 4. Escribir Cantidad_de_comandos 5. Para cada comando en Lista_de_comandos 6. Hacer 7. Escribir numeración 8. Escribir Cantidad_anclajes 9. Para cada anclaje en Lista_de_Anclajes 10. Hacer 11. Escribir valor_de_anclaje 12. Fin Para 13. Fin Para 14. Fin

4.2.2.*Cargar implante desde un archivo*

Este fragmento de código se encarga de leer el contenido de un archivo XML y llenar con los datos almacenados en este, un objeto de la clase Implant. Para llevar a cabo esta operación se deben leer correctamente las secciones del archivo XML. Primero se identifican los datos referente a la información textual del implante, luego se procede a leer las partes gráficas del implante de forma repetitiva dependiendo de la cantidad de secciones individuales que existan referente a las numeraciones.

MétodoGetImplanFromFile (Pseudocódigo)
1. Inicio 2. Crear LectorXML(dirección) 3. Crear ImplanteDigital 4. Leer Descipción desde LectorXML 5. Almacenar Descripción en ImplanteDigital 6. Leer Cantidad_Comandos

```
7. Mientras Cantidad_Comandos > 0
8. Hacer
9.  Leer Cantidad_anclajes
10. Mientras Cantidad_Anclajes > 0
11. Hacer
12. Leer Valor_anclaje
13. Almacernar Valor_Anclaje en ImplanteDigital
14. Decrementar Cantidad_Anclajes
15. Fin Mientras
16. Decrementar Cantidad_Comandos
17. Fin Mientras
18. Devolver ImplanteDigital
19. Fin
```

4.2.3.*Cargar las extensiones de la aplicación*

Estos fragmentos de código en su conjunto proveen la forma de leer y cargar las extensiones de la aplicación y permiten su posterior uso. Primero se extrae la llave pública con que se firma tanto la aplicación, como cada una de las extensiones para evitar inserciones foráneas, luego se localizan en una dirección en disco determinada todos los archivos con extensión .DLL. Se comprueban las llaves públicas de cada uno de los archivos detectados y se procede a agregar cada uno de estos a un catálogo, que luego compondrá sus partes en el contenedor de composición especificado el cual contendrá las extensiones de la aplicación listas para ser utilizadas.

Métodos para cargar las extensiones(Pseudocódigo)

```
1. Inicio
2. Crear Catálogo
3. Leer Archivos de Directorio(dirección)
4. Para cada Archivo
5. Hacer
```

```
6.  LLP = Extraer LLave_Pública de Archivo
7.  LLL = Extraer LLave_Pública del Sistema
8.  Si LLP = LLL
9.  Hacer
10. AV = Obtener Assembly del Archivo
11.  Agregar AV al Catálogo
12.  Fin Si
13. Fin Hacer
14. Catálogo.Componer
15. Fin
```

4.2.4. *Cargar implantes digitales ortopédicos*

Este fragmento de código permite dado un directorio del sistema de archivo y una serie de criterios de búsqueda, obtener el o los implantes digitales ortopédicos que coincidan con dichos criterios, para lo cual se carga cada implante digital, se obtienen los datos de su descripción y se comparan con los criterios de búsqueda, añadiéndose a una lista los resultados coincidentes.

Métodos para cargar los implantes digitales ortopédicos

```
1.  Inicio
2.  Crear Lista_Implantes
3.  Leer Archivos de Directorio(dirección)
4.  Para cada Archivo
5.  Hacer
6.   Descripción = Archivo.CargarDescripción
7.   Si Descripción.Contiene(criterios)
8.   Agregar Archivo a Lista_Implantes
9.   Fin Si
10. Fin Hacer
11. Devolver Lista_Implantes
12. Fin
```

En este capítulo se presentó el diagrama de componentes para la representación en componentes físicos; además fueron descritos algunos fragmentos importantes de la implementación del código del componente de software para brindar claridad y entendimiento a los desarrolladores que requieran hacer uso del mismo.

Conclusiones

Al evaluarse las tendencias actuales en la construcción de librerías de software se definió una estructura a modo de catálogo para la realización del componente de manipulación de los implantes digitales ortopédicos.

Del análisis de las librerías de implantes digitales ortopédicos se obtuvo que no existen estándares que definan la estructura, manipulación, visualización y almacenamiento de los implantes digitales ortopédicos; por lo que se propone el formato XML para archivar sus contenidos.

Con el análisis de las herramientas y tecnologías existentes se pudo identificar y garantizar que se contaran con las mejores alternativas para la implementación del sistema, mediante el uso de C# 4.0, Framework .Net 4.0 y Windows Presentation Foundation como plataforma moderna de desarrollo de interfaz de usuario.

Al evaluarse los procesos de negocio asociados a la gestión de implantes digitales ortopédicos se obtuvo un modelo guía para la implementación del sistema mediante la generación de los artefactos correspondientes a los flujos de trabajo propuestos por la metodología RUP, que sirvieron de base para el desarrollo del sistema.

El sistema se implementó de acorde a las pautas de diseño y lo establecido en la especificación de requisitos de software llevada a cabo. Esto garantiza la flexibilidad, robustez y el fácil mantenimiento de la arquitectura basada en el patrón MVVM, lo que permitió que la interfaz de usuario y la lógica del negocio se implementaran independientemente.

Recomendaciones

En aras de enriquecer la solución desarrollada se proponen las siguientes recomendaciones:

- Integrar el componente al proyecto planificador quirúrgico ortopédico permitiéndole a este último manipular y visualizar los implantes digitales ortopédicos, indispensables para la planificación preoperatoria.
- Desarrollar nuevos comandos gráficos para ser utilizados en el diseño de los implantes digitales ortopédicos, ampliando de esta manera las posibilidades de diseño de los mismos.
- Agregar nuevas herramientas de diseño y visualización al editor de implantes digitales ortopédicos dotándolo de nuevas funcionalidades que permitan una edición más flexible, rápida y eficiente.
- Investigar sobre algoritmos de encriptación y compresión, que permitan modificar el formato de almacenamiento seleccionado, sin afectar su flexibilidad, para dotarlo de una mayor seguridad ante ataques y reducir el tamaño en disco.

Referencias

1. Prensa Latina 28 de Marzo del 2006. *Vanguardia.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.vanguardia.co.cu/index.php?tpl=design/secciones/lectura/ciencia.tpl.html&newsid_obj_id=9950.

2. Hospital Ortopédico "Frank País". *Sermicuba.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.sermincuba.com.ar/FPais.htm.

3. Orthopedic Implants - Hip, Knee, Shoulder and Elbow. *Orthopedic Implants.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.implants.com/orthopedic_implants_surgery.html.

4. Biblioteca (Informática). *Academic.* [En línea] [Citado el: 10 de 12 de 2010.] http://www.esacademic.com/dic.nsf/eswiki/171371.

5. StudioLine Photo Classic. *http://www.studioline.biz.* [En línea] StudioLine. [Citado el: 1 de 3 de 2011.] http://www.studioline.biz/EN/products/overview-photo-classic/default.htm.

6. Zoner.com | Zoner Photo Studio PRO | Manage. *Zoner.com.* [En línea] Zoner. [Citado el: 1 de 3 de 2011.] http://www.zoner.com/ww-en/photo-studio-professional/organize.

7. Zoner.com | Zoner Photo Studio PRO | Edit. *Zoner.com.* [En línea] Zoner. [Citado el: 1 de 3 de 2011.] http://www.zoner.com/ww-en/photo-studio-professional/edit.

8. **Beneco.** Ashampoo Photo Commander 9.0.0 Final Gestion, Optimizacion y Edicion De Tus Fotografias. *FanaticoWarez.* [En línea] [Citado el: 15 de 04 de 2011.] http://www.fanaticowarez.org/2011/02/ashampoo-photo-commander-900-final.html.

9. Managed Extensibility Framework. *Codeplex.* [En línea] [Citado el: 13 de 1 de 2011.] http://mef.codeplex.com/wikipage?title=Overview&referringTitle=Home.

10. Definición de XML. *Mastermagazine.* [En línea] [Citado el: 13 de 1 de 2011.] http://www.mastermagazine.info/termino/7292.php.

11. XML. *Hipertexto.* [En línea] [Citado el: 13 de 1 de 2011.] http://www.hipertexto.info/documentos/xml.htm.

12. Metodologias de desarrollo. *Marvel Station.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.marblestation.com/?p=644.

13. Proceso de Desarrollo OpenUP. *Córdova Software Factory.* [En línea] [Citado el: 15 de 1 de 2011.] http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/.

14. IBM Rational Unified Process. *Grupo Soluciones Innova.* [En línea] [Citado el: 15 de 1 de 2011.] http://www.rational.com.ar/herramientas/rup.html.

15. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 01/01/2004.

16. Enterprise Architect. *SparxSystems.* [En línea] [Citado el: 14 de 1 de 2011.] http://www.sparxsystems.com.ar/products/ea.html.

17. .NET Framework 4. *MSDN.* [En línea] [Citado el: 16 de 1 de 2011.] http://msdn.microsoft.com/en-us/library/w0x726c2.aspx.

18. Que es subversion (SVN). *imapax.* [En línea] [Citado el: 16 de 1 de 2011.] http://www.imapax.com/soluciones/joomla/que-es-subversion-svn.html.

19. tortoisesvn. *Tigris.org.* [En línea] [Citado el: 16 de 1 de 2011.] http://tortoisesvn.tigris.org/.

20. Inversión de Control (IoC) – Inyección de dependencias (DI) | Lycka Bonita. *hachisvertas.* [En línea] [Citado el: 20 de 03 de 2011.] http://www.hachisvertas.net/blog/01/2008/03/27/inversion-de-control-ioc-inyeccion-de-dependencias-di.

21. Patrones Grasp « El Mundo Informático. *El Mundo Informático.* [En línea] [Citado el: 20 de 03 de 2011.] http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

22. CI-3711 Tópico Patrones de Diseño. *ldc.usb.ve.* [En línea] [Citado el: 21 de 03 de 2011.] http://ldc.usb.ve/~teruel/ci3711/patron3a/index.html.

23. uml2_sequencediagram. *www.sparxsystems.com.ar.* [En línea] sparxsystems. [Citado el: 24 de 03 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.

24. Sparx Systems - Tutorial UML 2 - Diagrama de Componentes:. *Sparx Systems.* [En línea] [Citado el: 20 de 04 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html.

Bibliografía

1. .NET Framework 4. *MSDN.* [En línea] [Citado el: 16 de 1 de 2011.] http://msdn.microsoft.com/en-us/library/w0x726c2.aspx.
2. An Introduction to Managed Extensibility Framework (MEF) - Part I - CodeProject. *CodeProject.* [En línea] The Code Project, 19 de 04 de 2011. [Citado el: 20 de 05 de 2011.] http://www.codeproject.com/KB/cs/mef_part1.aspx.
3. Aplicaciones ofimáticas. *Alfin EEES.* [En línea] [Citado el: 15 de 12 de 2010.] http://www.mariapinto.es/alfineees/ofimatica/como.htm.
4. **Beneco.** Ashampoo Photo Commander 9.0.0 Final Gestion, Optimizacion y Edicion De Tus Fotografias. *FanaticoWarez.* [En línea] [Citado el: 15 de 04 de 2011.] http://www.fanaticowarez.org/2011/02/ashampoo-photo-commander-900-final.html.
5. Biblioteca (Informática). *Academic.* [En línea] [Citado el: 10 de 12 de 2010.] http://www.esacademic.com/dic.nsf/eswiki/171371.
6. **Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson,Morgan Skinner.***C# 4 and .neT 4.* Indianapolis, Indiana : Wiley Publishing Inc., 2010. ISBN: 978-0-470-50225-9.
7. CI-3711 Tópico Patrones de Diseño. *ldc.usb.ve.* [En línea] [Citado el: 21 de 03 de 2011.] http://ldc.usb.ve/~teruel/ci3711/patron3a/index.html.
8. Definición de fichero. *Mastermagazine.* [En línea] [Citado el: 12 de 1 de 2011.] http://www.mastermagazine.info/termino/4996.php.
9. Definición de XML. *Mastermagazine.* [En línea] [Citado el: 13 de 1 de 2011.] http://www.mastermagazine.info/termino/7292.php.
10. Enterprise Architect. *SparxSystems.* [En línea] [Citado el: 14 de 1 de 2011.] http://www.sparxsystems.com.ar/products/ea.html.
11. **Griffiths, Chris Sells and Ian.***Programming WPF, Second Edition.* USA : O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2007. ISBN-10: 0-596-51037-3.

12. Hospital Ortopédico "Frank País". *Sermicuba.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.sermincuba.com.ar/FPais.htm.

13. IBM Rational Unified Process. *Grupo Soluciones Innova.* [En línea] [Citado el: 15 de 1 de 2011.] http://www.rational.com.ar/herramientas/rup.html.

14. Introducing Visual Studio. *MSDN.* [En línea] [Citado el: 16 de 1 de 2011.] http://msdn.microsoft.com/en-us/library/fx6bk1f4.aspx.

15. Inversión de Control (IoC) – Inyección de dependencias (DI) | Lycka Bonita. *hachisvertas.* [En línea] [Citado el: 20 de 03 de 2011.] http://www.hachisvertas.net/blog/01/2008/03/27/inversion-de-control-ioc-inyeccion-de-dependencias-di.

16. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.***El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 01/01/2004.

17. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.***El Proceso Unificado de Desarrollo de.* s.l. : PEARSON EDUCACIÓN S.A, 2000. ISNB 84-7829-036-2.

18. **Larman, C.***UML y Patrones: Introducción al análisis y programación orientada a objetos.* [PDF] Prentice Hall : s.n., 1999. MON-001311.

19. Managed Extensibility Framework - Home. *MSDN.* [En línea] Microsoft. [Citado el: 15 de 03 de 2011.] http://archive.msdn.microsoft.com/mef.

20. Managed Extensibility Framework. *Codeplex.* [En línea] [Citado el: 13 de 1 de 2011.] http://mef.codeplex.com/wikipage?title=Overview&referringTitle=Home.

21. Metodologias de desarrollo. *Marvel Station.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.marblestation.com/?p=644.

22. **Moser, Christian.** WPF Tutorial | Dependency Injection. *WPFTutorial.* [En línea] 2011. [Citado el: 12 de 05 de 2011.] http://www.wpftutorial.net/ReferenceArchitecture.html.

23. **Moser, Christian.** WPF Tutorial | Model-View-ViewModel Pattern. *WPFTutorial.* [En línea] 2011. [Citado el: 04 de 05 de 2011.] http://www.wpftutorial.net/MVVM.html.

24. Orthopedic Implants - Hip, Knee, Shoulder and Elbow. *Orthopedic Implants.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.implants.com/orthopedic_implants_surgery.html.

25. Patrones Grasp « El Mundo Informático. *El Mundo Informático.* [En línea] [Citado el: 20 de 03 de 2011.] http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

26. Portal: Salud en Cuba. *Ecured.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.ecured.cu/index.php/Portal:Salud_en_Cuba. 1.

27. Prensa Latina 28 de Marzo del 2006. *Vanguardia.* [En línea] [Citado el: 11 de 11 de 2010.] http://www.vanguardia.co.cu/index.php?tpl=design/secciones/lectura/ciencia.tpl.html&newsid_obj_id=9950.

28. **Pressman, Roger S.***Ingeniería del Software: un enfoque práctico. Parte I y II.* Madrid,2002, ed. 5ta : McGrawHill. MON-002581.

29. Proceso de Desarrollo OpenUP. *Córdova Software Factory.* [En línea] [Citado el: 15 de 1 de 2011.] http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/.

30. Que es subversion (SVN). *imapax.* [En línea] [Citado el: 16 de 1 de 2011.] http://www.imapax.com/soluciones/joomla/que-es-subversion-svn.html.

31. **Ramírez, Esmitt.***Planificación Preoperatoria Digital en Traumatología .* Caracas : Lecturas en Ciencias de la Computación, 2009-07. ISSN 1316-6239.

32. Sparx Systems - Tutorial UML 2 - Diagrama de Componentes:. *Sparx Systems.* [En línea] [Citado el: 20 de 04 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html.

33. StudioLine Photo Classic. *http://www.studioline.biz.* [En línea] StudioLine. [Citado el: 1 de 3 de 2011.] http://www.studioline.biz/EN/products/overview-photo-classic/default.htm.

34. tortoisesvn. *Tigris.org.* [En línea] [Citado el: 16 de 1 de 2011.] http://tortoisesvn.tigris.org/.

uml2_sequencediagram. *www.sparxsystems.com.ar.* [En línea] sparxsystems. [Citado el: 24 de 03 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.

35. Windows Presentation Foundation (WPF) - Home. *Codeplex.* [En línea] Codeplex Open Source Comunity, 2006-2011. [Citado el: 02 de 04 de 2011.] http://wpf.codeplex.com/.

36. WPF para patrón de diseño MVVM. *MSDN Magazine.* [En línea] Microsoft. [Citado el: 14 de 1 de 2011.] http://msdn.microsoft.com/es-es/magazine/dd419663.aspx.

37. WPF Tutorial | DelegateCommand. *WPFTutorial.* [En línea] 2011. [Citado el: 07 de 05 de 2011.] http://www.wpftutorial.net/DelegateCommand.html.

38. WPF Tutorial | Elegant way for INotifyPropertyChanged. *WPFTutorial.* [En línea] 2011. [Citado el: 06 de 05 de 2011.] http://www.wpftutorial.net/INotifyPropertyChanged.html

39. XML. *Hipertexto.* [En línea] [Citado el: 13 de 1 de 2011.] http://www.hipertexto.info/documentos/xml.htm.

40. Zoner.com | Zoner Photo Studio PRO | Edit. *Zoner.com.* [En línea] Zoner. [Citado el: 1 de 3 de 2011.] http://www.zoner.com/ww-en/photo-studio-professional/edit.

41. Zoner.com | Zoner Photo Studio PRO | Manage. *Zoner.com.* [En línea] Zoner. [Citado el: 1 de 3 de 2011.] http://www.zoner.com/ww-en/photo-studio-professional/organize.

Anexos

1. Anexos: Estructura del archivo XML que contiene el implante digital ortopédico

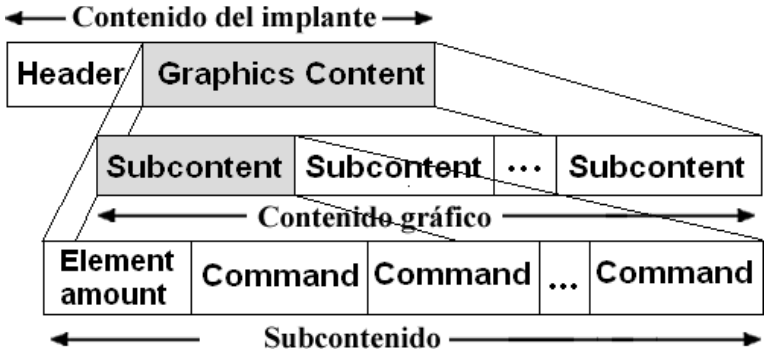


Figura 15 Estructura del archivo XML.

Header: Contiene la descripción del implante; nombre, familia, área, resolución y descripción textual.

Subcontent: Representan los subelementos del implante.

Command: Representan los comandos gráficos utilizados para dibujar el implante.

2. Anexos Interfaz de usuario del Studio Line Photo Classic

Figura 16. Interfaz de usuario del Studio Line Photo Classic

3. Anexos Interfaz de usuario del Zoner Photo Studio

Figura 17. Interfaz de usuario del Zoner Photo Studio

4. Anexos Ashampoo Photo Commander.

Figura 18. Interfaz de usuario del Ashampoo Photo Commander

5. Anexos Casos de Uso Expandidos.

CU # 1. Cargar Implante	
Objetivo	Cargar un implante dado criterios de búsqueda.
Actores	Planificador
Resumen	Permitir que el planificador suministre información para realizar una búsqueda y localizar el archivo en cuestión, y a su vez cargar la información de este.
Complejidad	Baja
Prioridad	Crítico
Referencias	RF 1.1
Precondiciones	No aplicable
Postcondiciones	No aplicable
Flujo de eventos	
Flujo básico: Cargar Implante	
1. El **Planificador** le brinda la información necesaria sobre los criterios de búsqueda al sistema. 2. El **Sistema** busca el archivo correspondiente a esas características y procede a cargarlo. 3. El Sistema brinda información para dibujar el implante. 4. Termina el caso de uso.	
Requisitos no funcionales	RNDI 1, RNDI 2, RNDI 3, RNDI 4
Prototipo de interfaz	Anexo X

CU # 2. Dibujar Implante

Objetivo	Dibujar un implante en un contexto de imagen.
Actores	Planificador
Resumen	Un conjunto de acciones y transformaciones necesarias para que el planificador pueda visualizar el implante.
Complejidad	Media
Prioridad	Crítico
Referencias	RF 2.1, RF 2.2, RF 2.3
Precondiciones	No aplicable
Postcondiciones	No aplicable
Flujo de eventos	
Flujo básico: Dibujar Implante	
1. El **Planificador** brinda el id de la sub-imagen, ángulo de rotación, la posición y el contexto de dibujo. 2. El **Sistema** crea una matriz de rotación y de traslación. 3. El **Sistema** le aplica las matrices de rotación y traslación a la sub-imagen seleccionada y la dibuja sobre el contexto de dibujo. 4. Termina el caso de uso.	
Requisitos no funcionales	RNDI 1, RNDI 2, RNDI 3, RNDI 4
Asuntos pendientes	

CU # 3. Gestionar diseño	
Objetivo	Diseñar un implante digital ortopédico.
Actores	Diseñador

Resumen	El planificador le brinda criterios de búsqueda dado los cuales el sistema encuentra el archivo correspondiente al implante indicado y carga la información correspondiente al mismo.
Complejidad	Baja
Prioridad	Crítico
Referencias	RF 3.1
Precondiciones	No aplicable
Postcondiciones	No aplicable
Flujo de eventos	
Flujo básico: Crear diseño	
1. El **Diseñador** selecciona la opción crear diseño. 2. El **Sistema** crea un diseño vacío. 3. Termina el caso de uso.	
Flujo básico: Almacenar diseño	
1. El **Diseñador** selecciona la opción crear diseño. 2. El **Sistema** brinda la opción de seleccionar una ubicación en disco para la creación del archivo que contendrá la información del implante 3. El **Diseñador** introduce la dirección y nombre del archivo. 4. El S**istema** almacena la información del implante en el archivo previamente creado en formato XML. 5. Termina el caso de uso.	
Flujos alternos	
3a. Nombre de archivo existente	
1. El **Sistema** muestra información de que existe un fichero con ese nombre y brinda la opción de sobrescribirlo.	
Flujo básico: Suprimir diseño	

1. El **Diseñador** selecciona la opción suprimir diseño 2. El S**istema** elimina el diseño y toda la información referente al mismo. 3. Termina el caso de uso.

Requisitos no funcionales	RNIU 1, RNDI 1, RNDI 2, RNDI 3, RNDI 4
Asuntos pendientes	Posibles mejoras al caso de uso.

CU # 4. Editar implante	
Objetivo	Editar el contenido del implante
Actores	Diseñador
Resumen	Conjunto de operaciones necesarias para editar la información del implante y diseñar su representación visual.
Complejidad	Alta
Prioridad	Crítico
Referencias	RF 3.2, RF 3.3
Precondiciones	No aplicable
Postcondiciones	No aplicable
Flujo de eventos	
Flujo básico: Editar información	
1. El **Diseñador** introduce la información referente al implante. 2. El **Sistema** guarda temporalmente dicha información para un posterior almacenamiento en archivo. 3. Termina el caso de uso.	
Flujo básico: Insertar comando gráfico	

1. El **Diseñador** selecciona un comando gráfico a adicionar al diseño. 2. El **Sistema** brinda puntos de anclaje para permitir alterar las dimensiones y posición del comando gráfico. 3. Termina el caso de uso.
Flujo básico: Eliminar comando gráfico
2. El **Diseñador** selecciona el comando gráfico a eliminar. 3. El **Sistema** remueve el comando gráfico y su representación visual del área de dibujo del diseño. 4. Termina el caso de uso.

Requisitos no funcionales	RNIU 1, RNDI 1, RNDI 2, RNDI 3, RNDI 4
Asuntos pendientes	

CU # 5. Pre-visualizar diseño	
Objetivo	Pre-visualizar un implante en edición
Actores	Diseñador
Resumen	Un conjunto de operaciones necesarias para pre-visualizar el diseño de un implante.
Complejidad	Media
Prioridad	Secundario
Referencias	RF 3.4
Precondiciones	No aplicable
Postcondiciones	No aplicable
Flujo de eventos	

Flujo básico: Pre-visualizar implante	
1. El **Diseñador** selecciona la opción de pre-visualización. 2. El **Sistema** crea una nueva ventana con la representación del diseño actualmente en edición. 3. El **Sistema** brinda 2 puntos de anclajes para determinar posición y ángulo de rotación para dibujar el implante. 4. Termina el caso de uso.	
Requisitos no funcionales	RNIU 1, RNDI 1, RNDI 2, RNDI 3, RNDI 4
Asuntos pendientes	

6. Anexos Diagrama de Clases

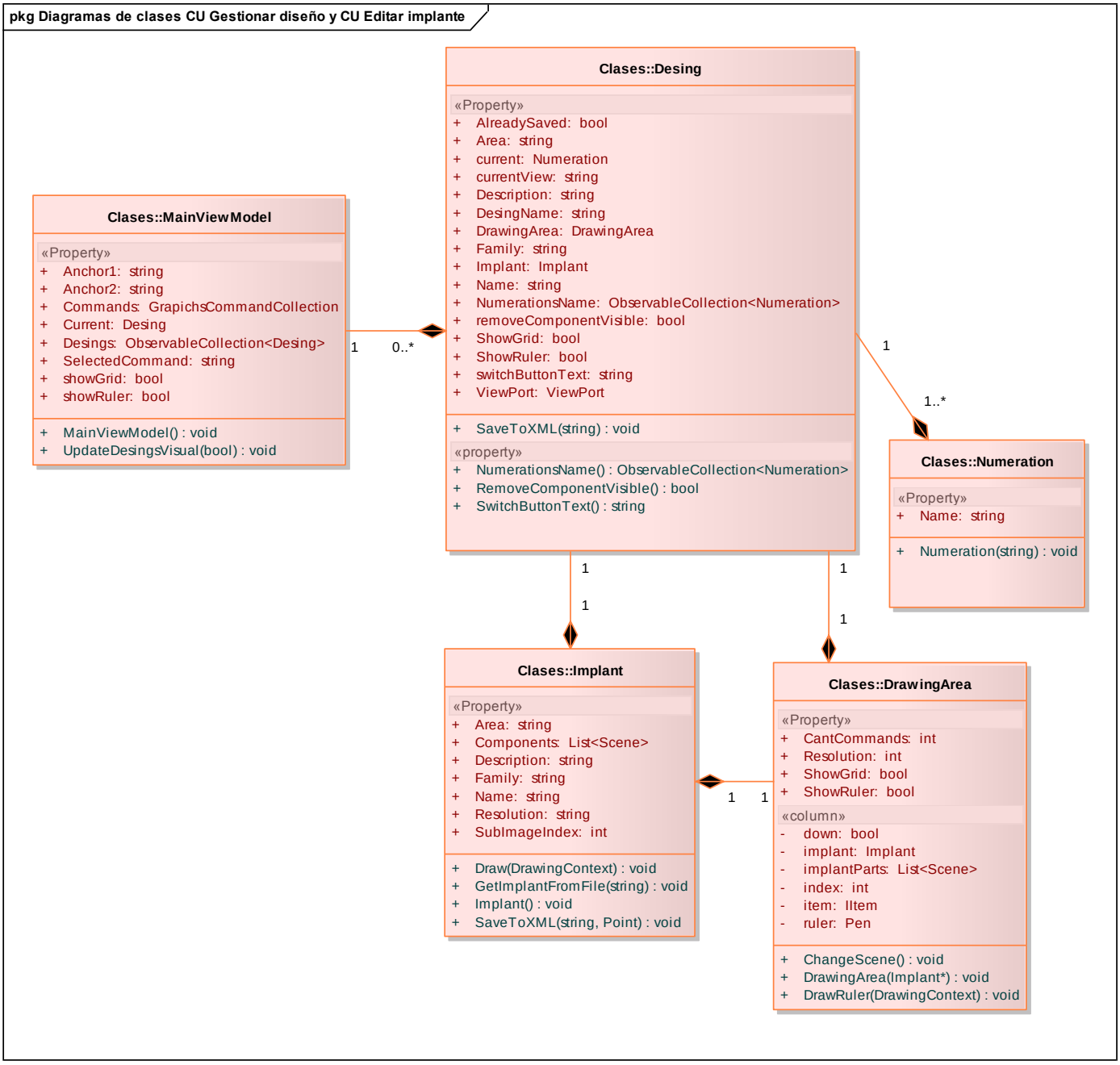


Figura 19. Diagrama de Clases: CU Gestionar diseño

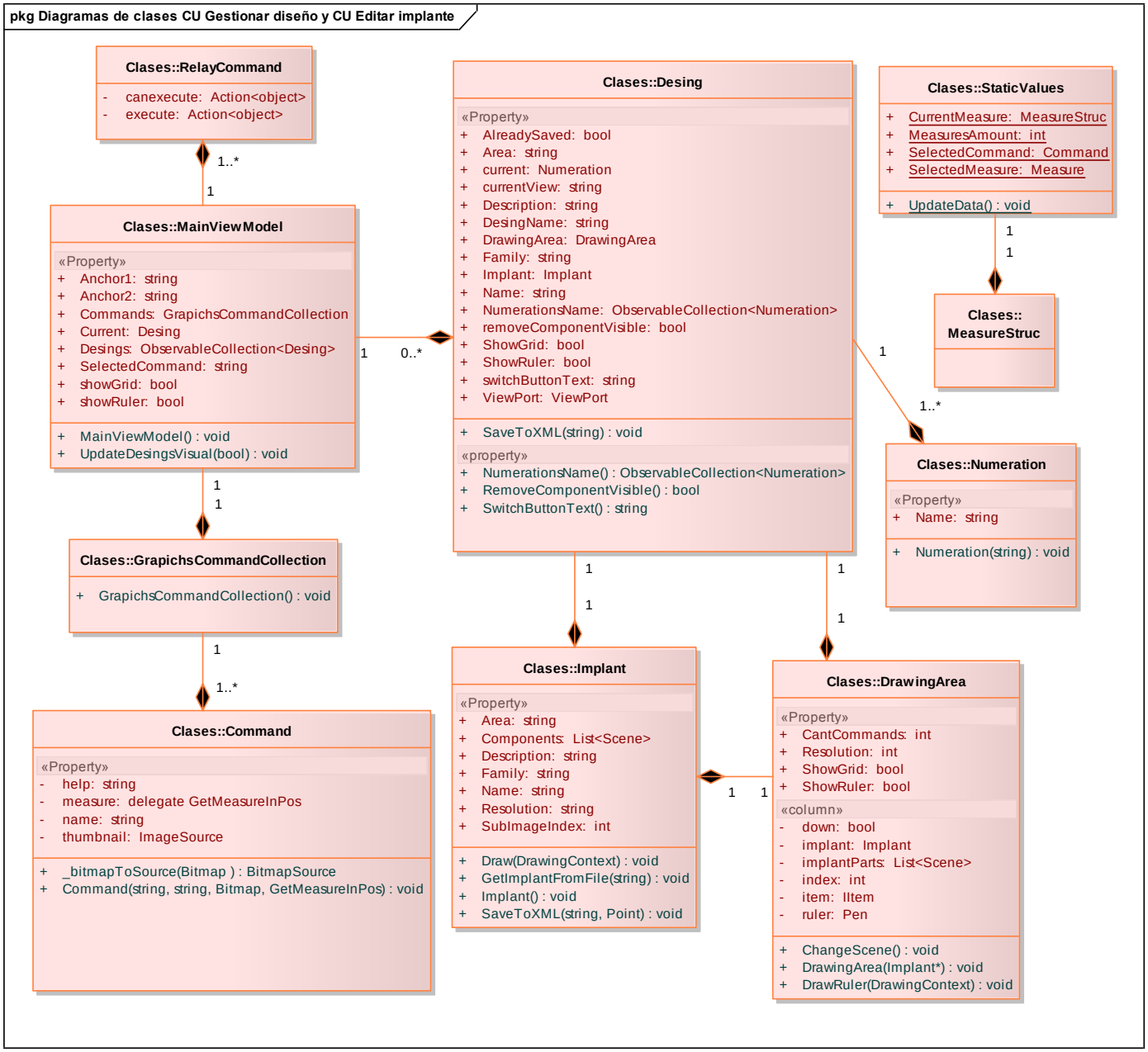


Figura 20. Diagrama de Clases: CU Editar implante.

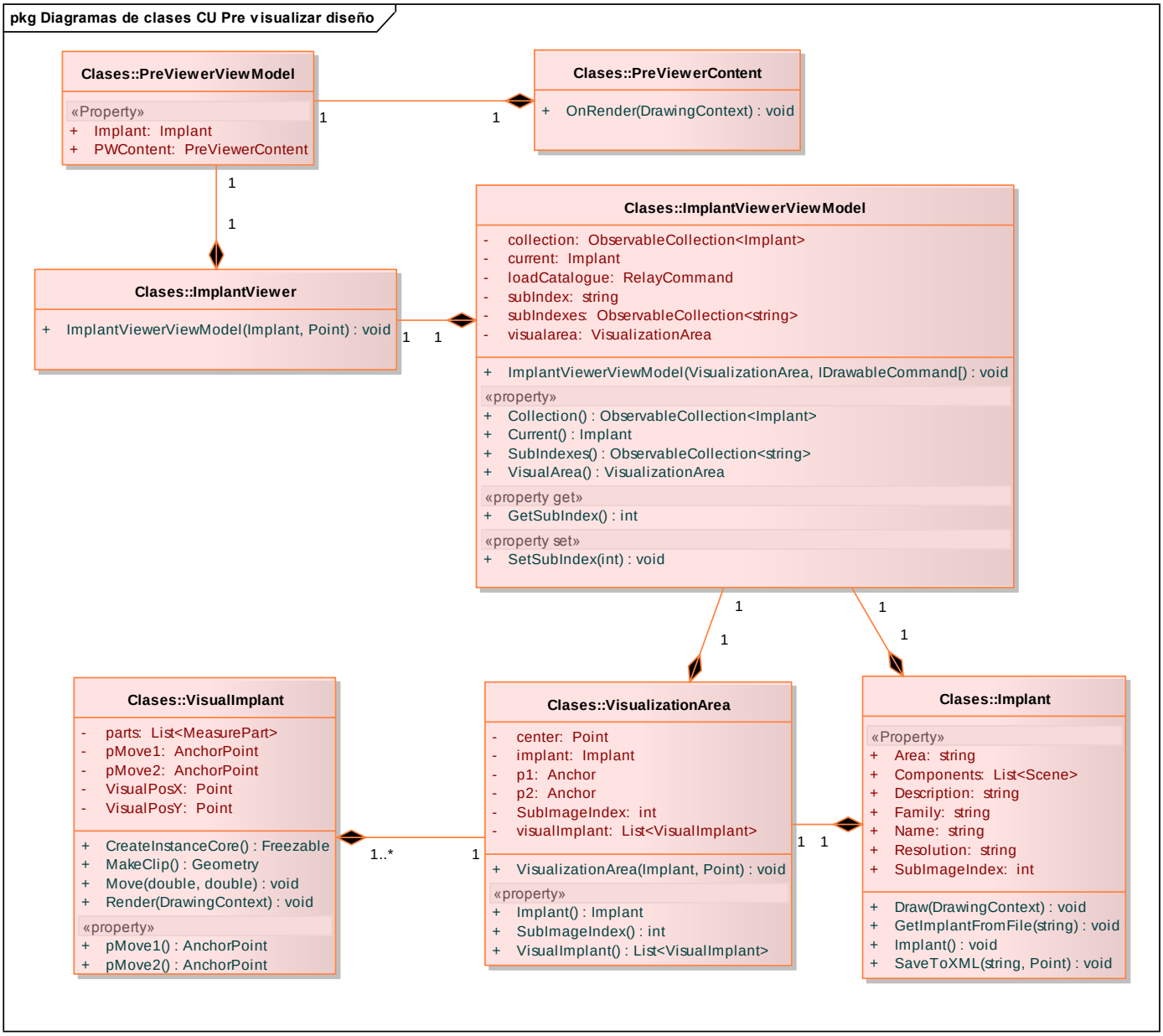


Figura 21. Diagrama de Clases: CU Pre-visualizar diseño.

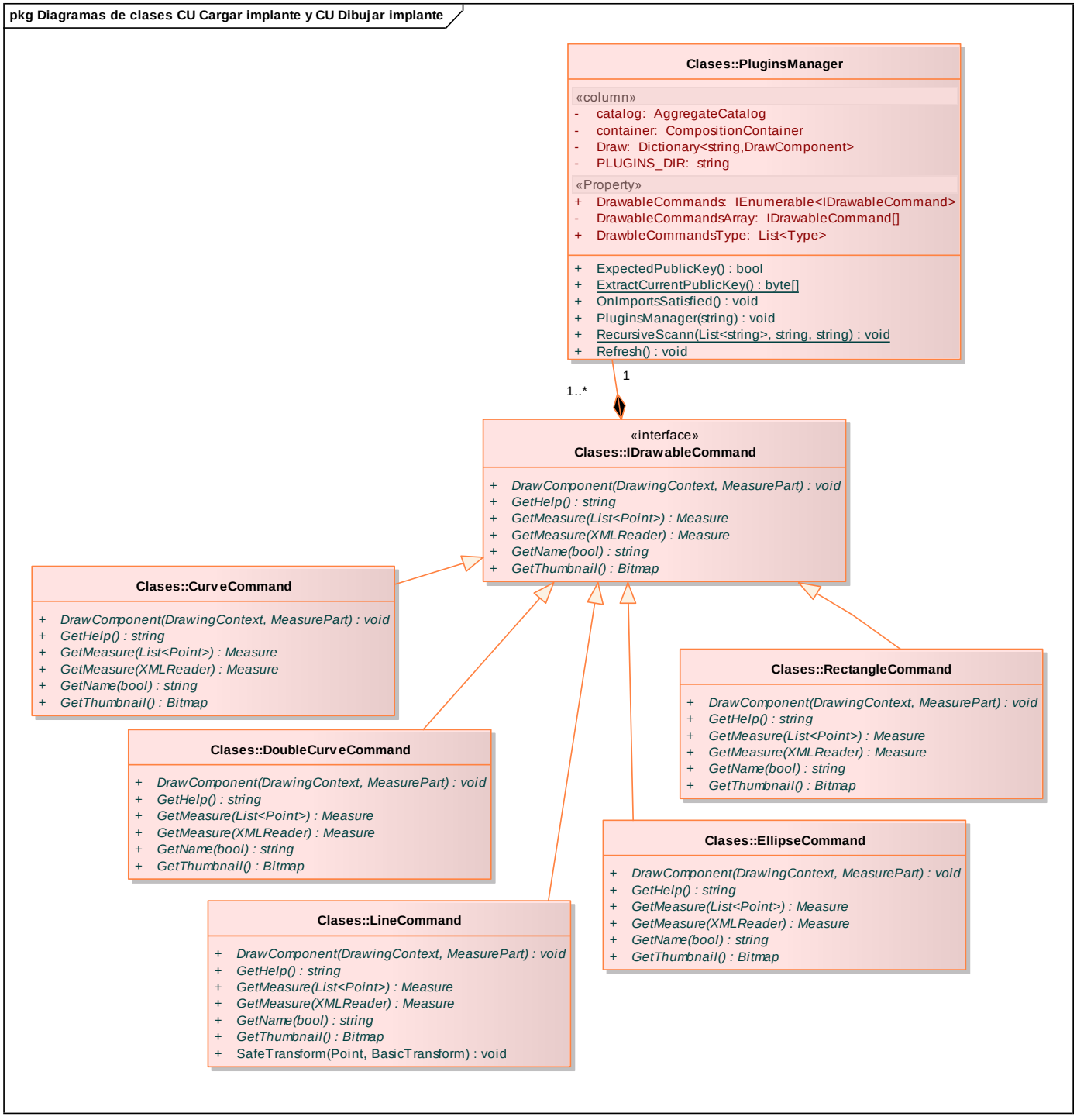


Figura 22. Diagrama de Clases: CU Dibujar Implante.

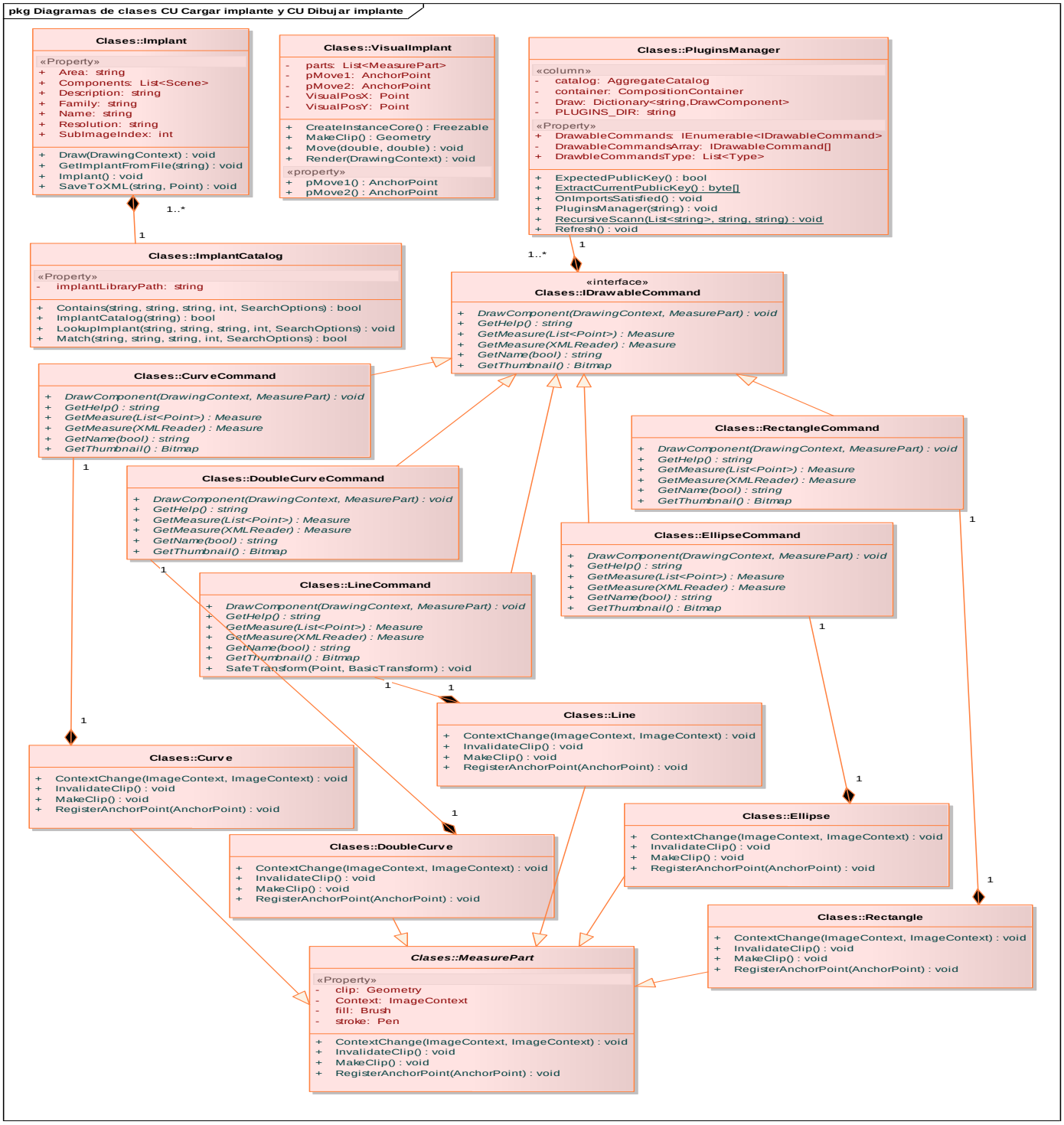


Figura 23. Diagrama de Clases: CU Cargar implante.

7. Anexos Diagrama de secuencia: Editar información referente al implante.

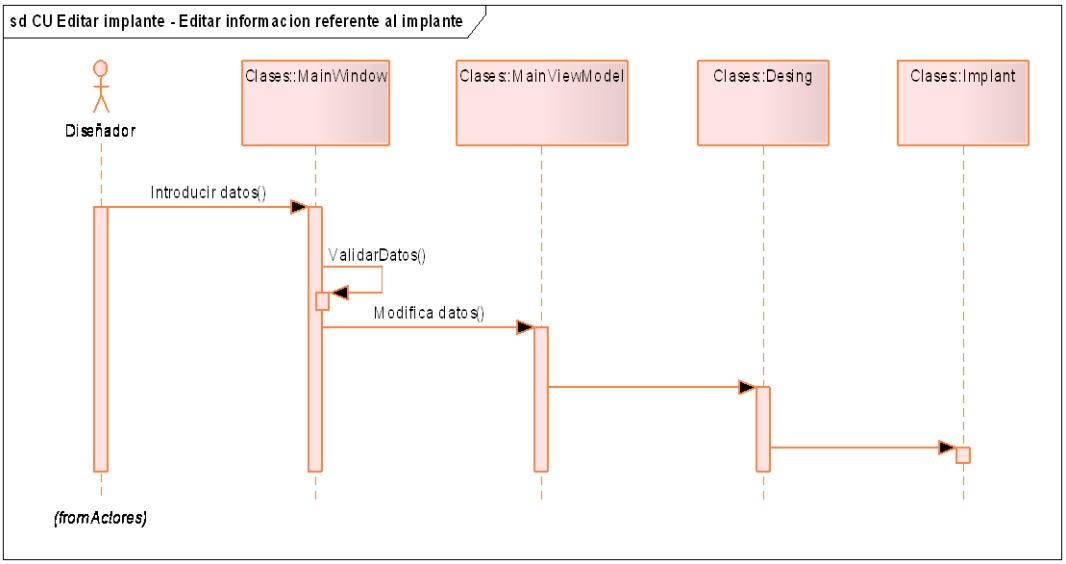


Figura 24. Diagrama de secuencia: Editar información referente al implante.

8. Anexos Diagrama de secuencia: Editar representación visual.

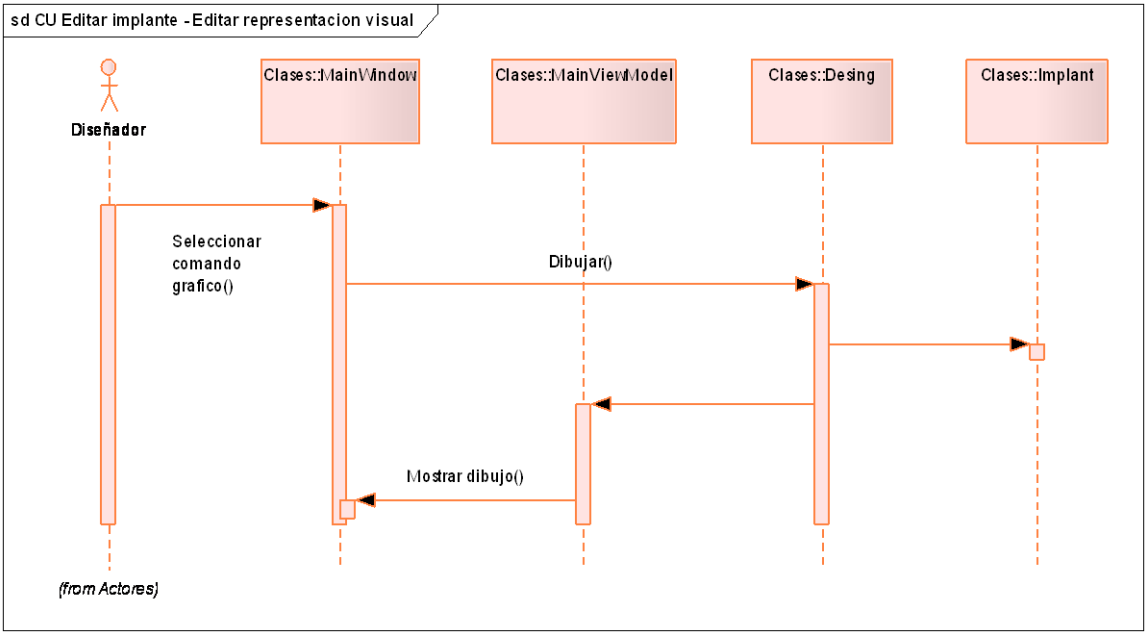


Figura 25. Diagrama de secuencia: Editar representación visual.

9. Anexos Diagrama de secuencia: Crear diseño

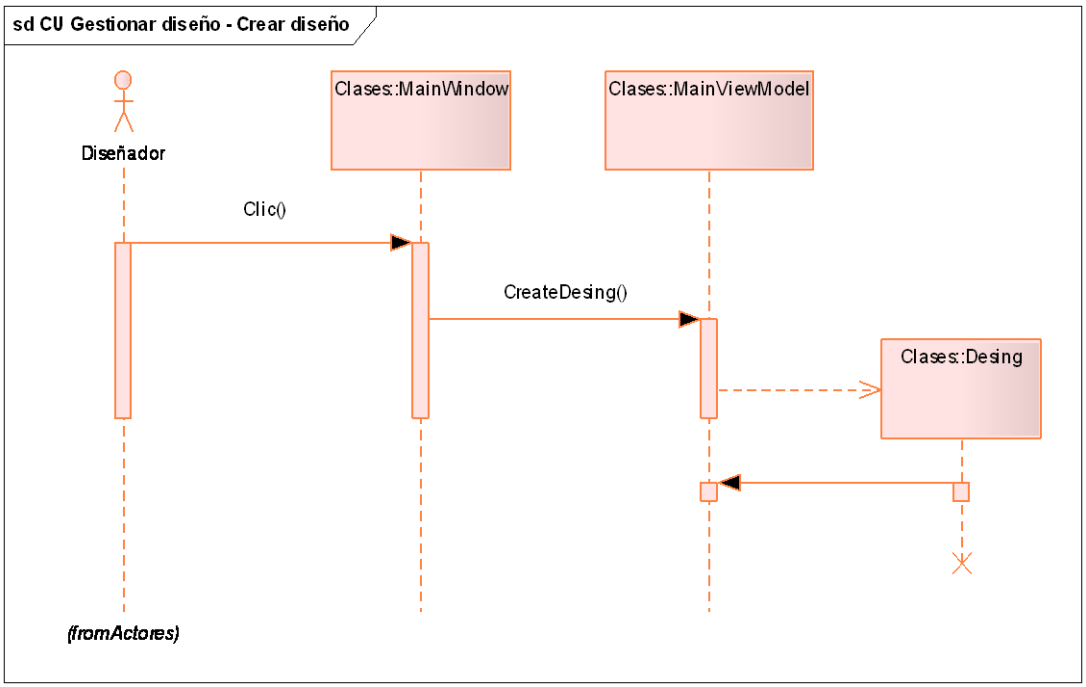


Figura 26. Diagrama de secuencia: Crear diseño.

10. Anexos Diagrama de secuencia: Suprimir diseño.

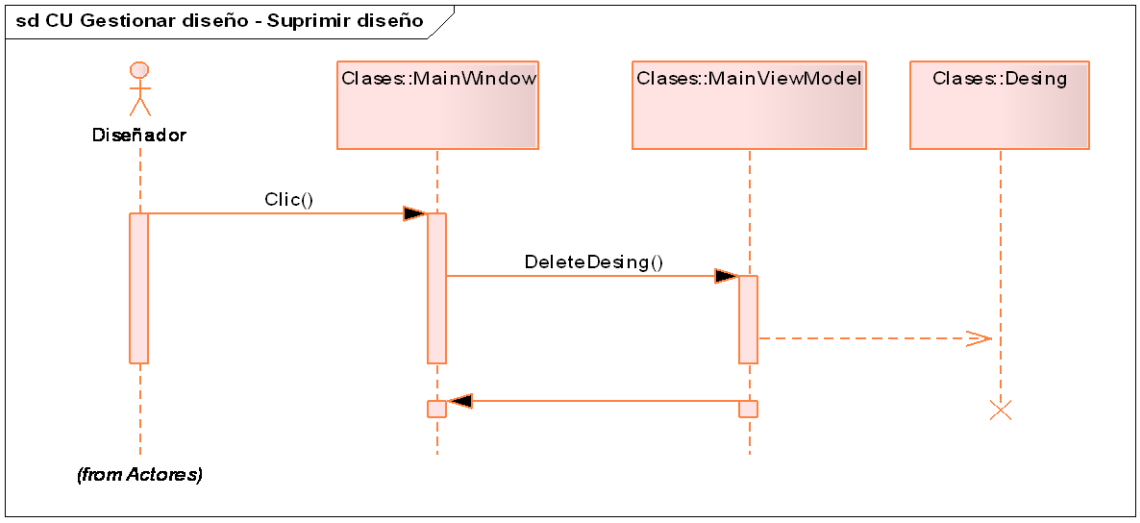


Figura 27. Diagrama de secuencia: Suprimir diseño.

11. Anexos Diagrama de secuencia: Almacenar diseño.

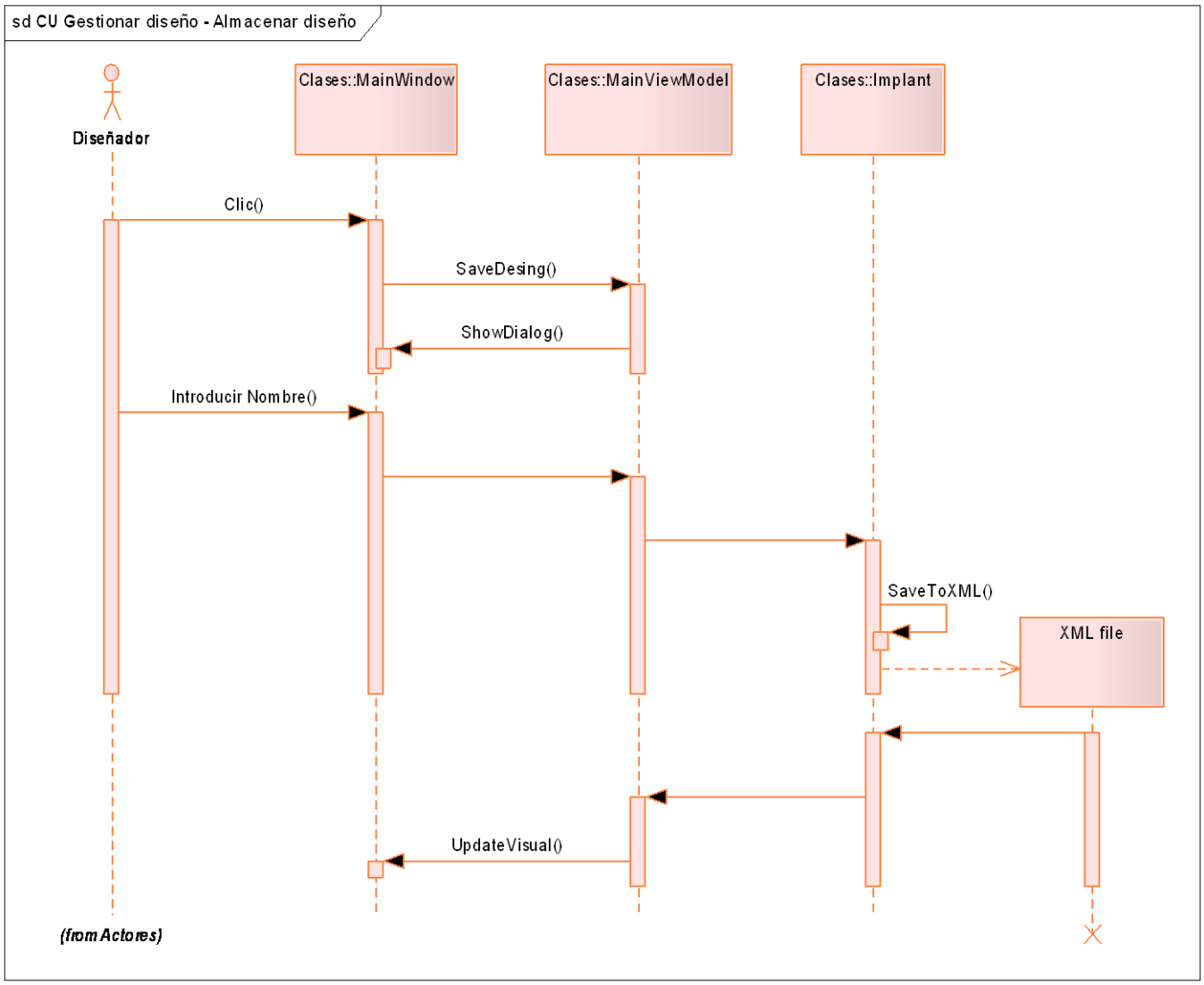


Figura 28. Diagrama de secuencia: Almacenar diseño.

12. Anexos Diagrama de secuencia: Pre-visualizar diseño.

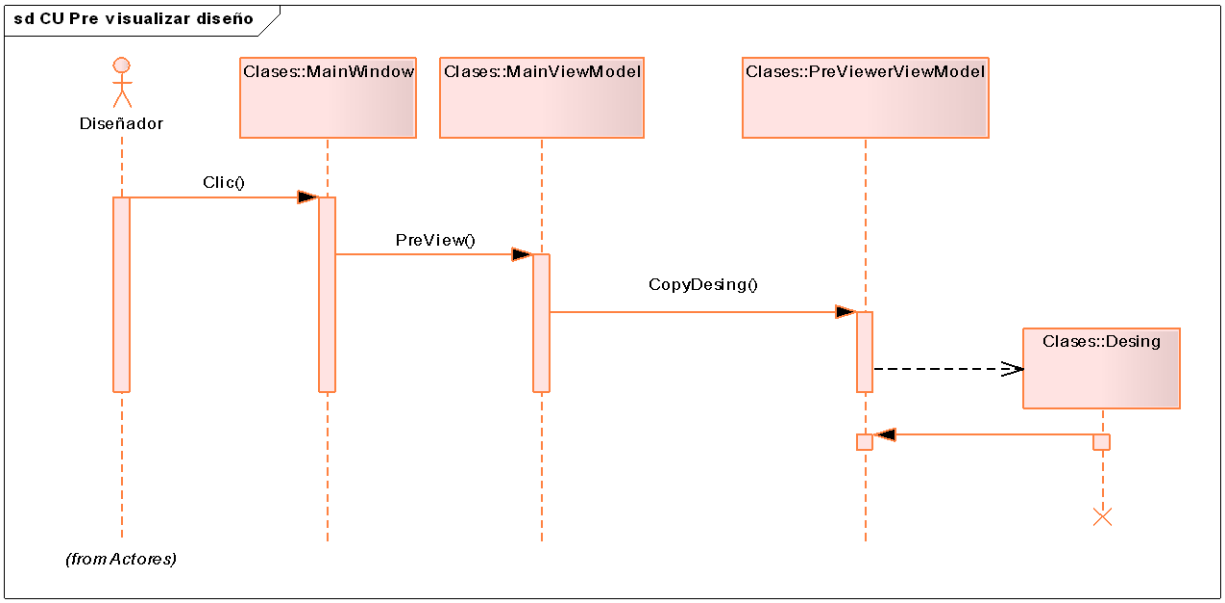


Figura 29. Diagrama de secuencia: Pre-visualizar diseño.

13. Anexos Diagrama de secuencia: Cargar implante.

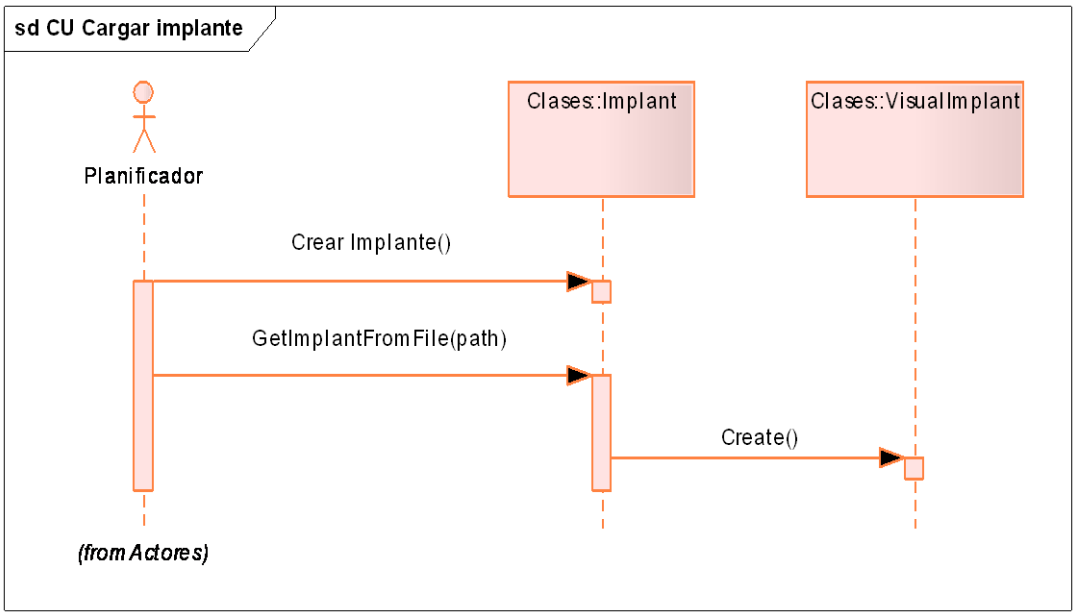


Figura 30. Diagrama de secuencia: Cargar implante.

14. Anexos Diagrama de secuencia: Dibujar implante.

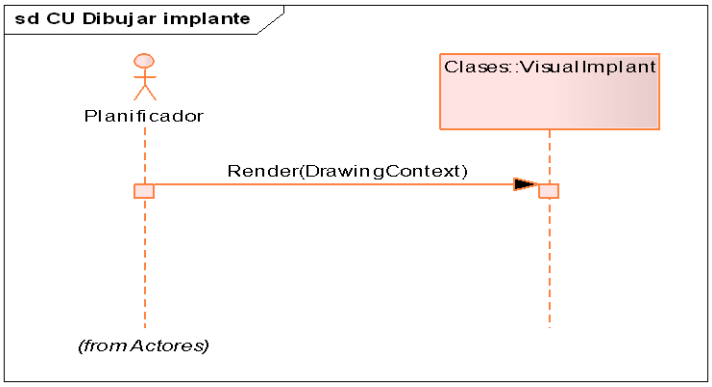


Figura 31. Diagrama de secuencia: Dibujar implante.

15. Anexos Interfaz del Editor de implantes digitales ortopédicos.

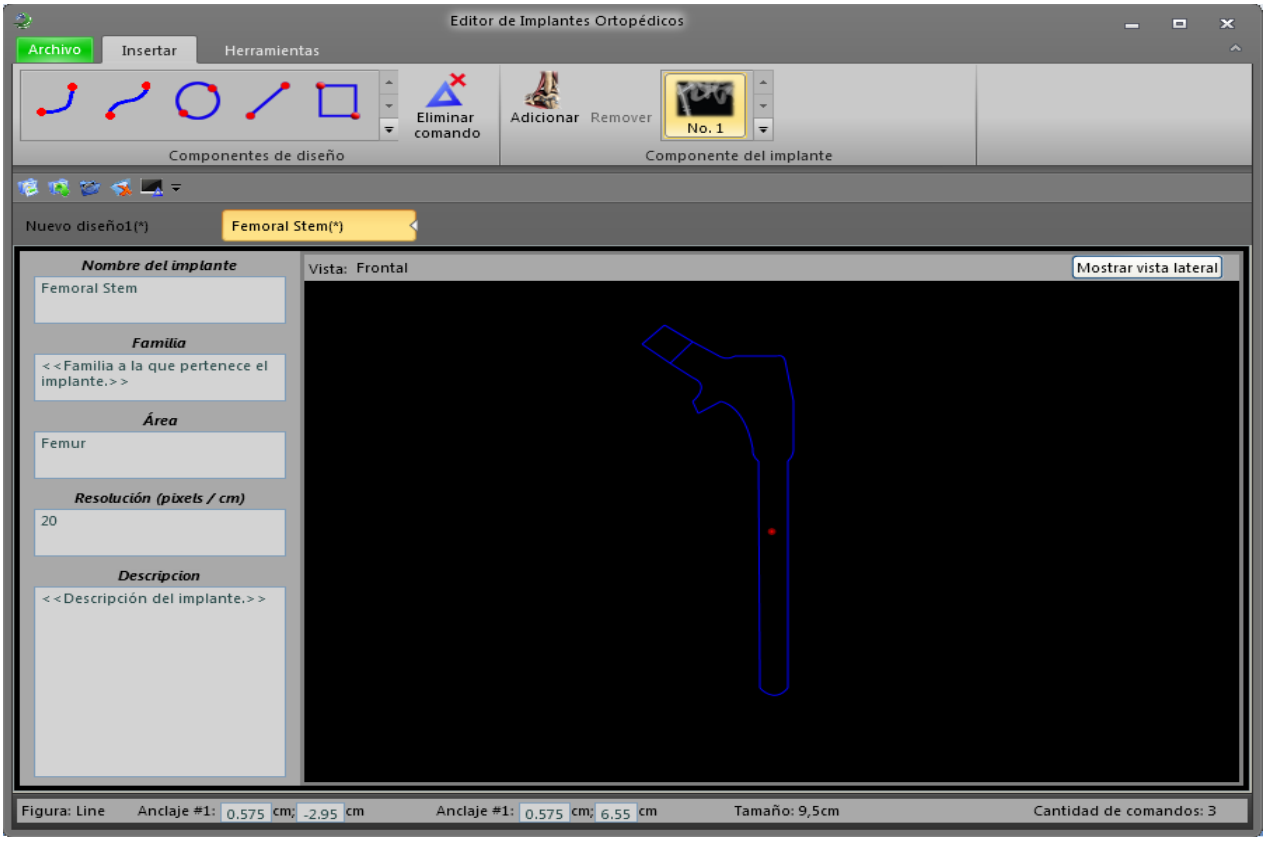


Figura 32. Interfaz del Editor de implantes digitales ortopédicos

Glosario de términos

Acetábulo: Es, en anatomía, la porción articular cóncava de la superficie de la pelvis, formada por el ilion, el isquion y el pubis, a la cual se articula la cabeza del fémur, formando la articulación de la cadera.

ADO.NET: Es un conjunto de componentes del software que pueden ser usados por los programadores para acceder a datos y a servicios de datos. Es una parte de la biblioteca de clases base que están incluidas en el Microsoft .NET Framework. Es comúnmente usado por los programadores para acceder y para modificar los datos almacenados en un Sistema Gestor de Bases de Datos Relacionales, aunque también puede ser usado para acceder a datos en fuentes no relacionales.

ASP.NET: Es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de 2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime.

BPMN: Bussiness Process Modeling Notation. Es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (workflow). BPMN fue inicialmente desarrollada por la organización Business Process Management Initiative (BPMI), y es actualmente mantenida por el OMG (Object Management Group), luego de la fusión de las dos organizaciones en el año 2005. Su versión actual, a abril de 2011, es la 2.0.

CESIM: Centro de software imagenológico

CVS: Concurrent Versions System o simplemente CVS, también conocido como Concurrent Versioning System, es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren. CVS se ha hecho popular en el mundo del software libre. Sus desarrolladores difunden el sistema bajo la licencia GPL.

DICOM: Digital Imaging and Communications in Medicine, estándar reconocido mundialmente para el intercambio de imágenes médicas.

Framework: define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

GPS: Sistema de Posicionamiento Global (Global Positioning System). Este sistema permite la localización de algún objeto (persona, nave, vehículo) en cualquier punto del planeta. El sistema GPS utiliza triangulación para poder calcular la posición de ese objeto.

Implante ortopédico: Es un material sintético o natural que es introducido en el cuerpo con la intención de sanar, curar o corregir algún problema de salud. Los diseños de implantes ortopédicos son certificados usando la norma de calidad ISO 9001.

JPEG: Joint Photographic Experts Group. Es la norma de compresión de imágenes más utilizada a nivel mundial.

LDAP: Son las siglas de ***L****ightweight* ***D****irectory* ***A****ccess* ***P****rotocol* (en español *Protocolo Ligero de Acceso a Directorios*) que hacen referencia a un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también es considerado una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

MINSAP: Ministerio de Salud Pública. Es el organismo rector del Sistema Nacional de Salud, encargado de dirigir, ejecutar y controlar la aplicación de la política del Estado y del gobierno cubano en cuanto a la Salud pública,

Osteoartritis: También llamada enfermedad articular degenerativa que ocurre cuando el cartílago se rompe o desgasta como consecuencia de exceso de peso corporal y/o la falta de movimiento de la articulación.

Osteomeoarticular: Conjunto de órganos que realiza la función de locomoción.

PAM: *Pluggable Authentication Modules.* Utiliza una arquitectura conectable y modular, que otorga al administrador del sistema de una gran flexibilidad en establecer las políticas de autenticación para el sistema. En la mayoría de los casos, el archivo de configuración por defecto PAM para una aplicación tipo PAM es suficiente.

RAW: El formato de imágenes RAW (traducción de "No Alterado" o "Forma Natural"; en el caso de las imágenes, entiéndase como "Formato de Imagen sin Compresión" o "Crudo") es un formato de archivo digital de imágenes que contiene la totalidad de los datos de la imagen tal y como ha sido captada por el sensor digital de la cámara fotográfica.

SGML: Standard Generalized Markup Language. Lenguaje de Anotaciones Generales. Es un estándar internacional. SGML es un metalenguaje, o sea un lenguaje usado para describir otros lenguajes.

SQL: El lenguaje de consulta estructurado o SQL (por sus siglas en inglés *structured query language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre ella.

Sub-imagen: Identifica una representación gráfica de las cuales está compuesto el implante digital ortopédico.

UML: Unified Modeling Language. Es una de las mejores herramientas para analizar y diseñar sistemas de software. Permite modelar (analizar y diseñar) sistemas orientados a objetos.

WWW: World Wide Web. Es un sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet. Con un navegador web, un usuario visualiza sitios web compuestos de páginas web que pueden contener texto, imágenes, videos u otros contenidos multimedia, y navega a través de ellas usando hiperenlaces.

XML: Es una versión de SGML, diseñado especialmente para los documentos de la web. Permite que los diseñadores creen sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones